Università della Calabria

Dipartimento di Matematica e Informatica

Dottorato di Ricerca in Matematica e Informatica

XXXVI ciclo

Tesi di Dottorato

A Logic-based Framework for Characterizing Nexus of Similarity within Knowledge Bases

Settore Scientifico Disciplinare INF/01 – INFORMATICA

Coordinatore: Ch.mo Prof. Giorgio Terracina

Supervisore: Ch.mo Prof. Marco Manna

Dottorando: Dott. Aldo Ricioppo

A Logic-based Framework for Characterizing Nexus of Similarity within Knowledge Bases

Aldo Ricioppo
Department of Mathematics and Computer Science, UNICAL

April 4, 2024

Sommario

Le questioni complesse spesso richiedono il riconoscimento di connessioni significative tra diverse entità. Ad esempio, selezionare il CV ottimale, trovare un appartamento adatto o decifrare i fattori causali dietro un disturbo specifico sono solo alcuni esempi, e ciascuno di essi richiede una profonda comprensione di ciò che caratterizza un insieme di entità.

Oltre agli esempi sopra menzionati, dover riconoscere le somiglianze tra le entità è un fenomeno ricorrente in una moltitudine di scenari del mondo reale. Ricercatori provenienti da diversi campi hanno presentato varie metodologie nel corso dell'ultimo secolo per valutare queste somiglianze, comunemente chiamate *similarity*. In tempi recenti, lo slancio è aumentato con l'avvento dei "Google Sets", portando al fervido sviluppo di strategie per amplificare un dato insieme di entità preservando le loro condivise e originali proprietà interconnesse.

Di conseguenza, le metodologie attuali tengono in considerazione, in un modo o nell'altro, proprietà che siano al contempo rilevanti e interconnesse condivise da entità, un concetto che chiamiamo *nexus of similarity*. Le macchine hanno dimostrato una notevole abilità nel gestire valutazioni di somiglianza, spesso restituendo come risultato punteggi numerici o impostando espansioni dell'input iniziale, dando così all'utente finale l'opportunità di osservare entità simili a quelle che stava cercando. Tuttavia, c'è un notevole divario nel *caratterizzare* (*characterizing*) formalmente il nexus of similarity in un modo che sia intelligibile dalle macchine e interpretabile dall'intelletto umano, soprattutto considerando che i tentativi che hanno guadagnato maggiore successo finora sono spesso legati a casi molto specifici, come quelli realizzati rispetto ai grafi RDF.

Per colmare queste lacune, il nostro lavoro contribuisce in modo significativo alla letteratura esistente. Il nostro obiettivo è costruire un nuovo framework basato su costrutti logici, progettato per delineare sistematicamente e autonomamente il nexus of similarity. Il nostro framework si estende non solo a coppie di entità ma anche a insiemi di tuple di entità, che chiamiamo *relazioni anonime* (*anonymous relations*), all'interno di una

base di conoscenza.

Inoltre, la nostra analisi comprende un esame approfondito della complessità computazionale inerente al framework proposto. Tale indagine consente una visione approfondita della sua fattibilità e una successiva valutazione della scalabilità. Entrambi sono componenti critici per l'applicazione pratica del framework.

In sintesi, il nostro studio apre la strada a un nuovo approccio basato sulla conoscenza in grado di caratterizzare il nexus of similarity e che può essere utilizzato come mezzo per eseguire l'espansione dell'insieme di entità in modo chiaro e comprensibile per gli esseri umani. Due dei principali componenti integrali del nostro framework saranno le risorse semantiche, specificamente le *basi di conoscenza selettive* (*selective knowledge bases*), che nella loro essenza sono basi di conoscenza dotate di un particolare algoritmo supplementare, e i *linguaggi di spiegazione* (*explanation languages*), di cui noi ne prenderemo in considerazione uno in particolare, che a nostro avviso possiede tutte le caratteristiche ideali per essere considerato un linguaggio idoneo a rivelare al meglio il nexus of similarity. Durante il lavoro giustificheremo anche le nostre scelte di design relative al framework. Con l'ausilio di questi mezzi miriamo a colmare la lacuna percepibile nella caratterizzazione dei nexus of similarity. Allo stesso tempo, mostreremo come alcuni degli attuali approcci all'espansione degli insiemi di entità (entity set expansion) non tengono in considerazione che per loro stessa natura questo tipo di espansioni dovrebbe assumere la forma di una tassonomia piuttosto che di una catena. Per risolvere quest'altra lacuna, introdurremo il concetto di *grafo di espansione* (*expansion graph*).

Abstract

Complex challenges frequently necessitate the establishment of significant connections among diverse entities. For instance, selecting the optimal CV, finding a suitable apartment or deciphering the causal factors behind a specific disorder are just a few examples, and each of them requires deep understanding of what characterizes a set of entities.

In addition to the aforementioned examples, having to recognize similarities between entities is a recurring phenomenon in a multitude of real-world scenarios. Researchers from different fields have presented various methodologies over the last century to evaluate these similarities, commonly called *similarity*. In recent times, momentum has increased with the advent of "Google Sets", leading to the fervent development of strategies to amplify a given set of entities while preserving their original shared interconnected properties.

As a consequence, current methodologies can encompass, in a way or another, relevant interconnected properties shared by entities, a concept we term the *nexus of similarity*. Machines have demonstrated considerable prowess in handling similarity evaluations, often returning numerical scores as a result, and set expansions, thus giving the end user the opportunity to observe entities similar to those he was looking for. However, there is a notable gap in formally *characterizing* the nexus of similarity in a way that is intelligible by machines and interpretable by human intellect, especially considering that the attempts that have gained the most traction thus far are often bound to cases very specific, such as those made with respect to RDF graphs.

To address these gaps, our endeavor contributes significantly to the existing literature. We aim to construct a novel framework grounded in logical constructs, designed to systematically and autonomously delineate the nexus of similarity. Our framework extends not only to pairs of entities but also to sets of tuples of entities, which we term *anonymous relations*, within a knowledge base.

Furthermore, our analysis encompasses an in-depth examination of the

computational complexity inherent in the proposed framework. Such an investigation affords a thorough insight into its feasibility and a subsequent evaluation of scalability. Both are critical components for the framework's practical application.

In summary, our study pioneers a novel, knowledge-driven approach capable of characterizing nexus of similarity and that can be used as a means to perform entity set expansion in a manner clear and intelligible to humans. Two of the principal components integral to our framework will be the semantic resources, specifically *selective knowledge bases*, that in their essence are knowledge bases equipped with a particular supplementary algorithm, and the *explanation languages*, of which we will take into consideration one in particular, which in our opinion has all the ideal characteristics to be considered a language suitable to reveal the nexus of similarity as best as possible. During the work, we will also justify our design choices. With the help of these means we aim to fill the perceptible gap in the characterization of nexus of similarity. At the same time, we will show how some of the current approaches to entity set expansion do not notice that by their very nature this kind of expansions should take the form of a taxonomy rather than a chain. To resolve this other gap, we will introduce the concept of *expansion graph*.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

One of the most prevalent cognitive activities that individuals routinely perform is the process of discerning similarities and differences among collections of objects or entities, guided by various attributes such as shape, size, color, and function. These *shared interconnected properties* shall henceforth be referred to as the *nexus of similarity*. The capability to juxtapose entities and extract sophisticated characterizations is quintessential for comprehending the world, acquiring new knowledge, making informed decisions, and stimulating creativity.

People routinely compare a wide array of objects, including fruits, cars, books, and clothes, for various purposes. Whether classifying them into categories, evaluating their qualities, making decisions, or solving problems, this act of comparison guides choices in everyday life. For instance, at a supermarket, individuals may compare fruits based on appearance, taste, price, or nutritional value to select the best option for their needs.

Beyond tangible objects, people also compare intangible entities like goods, brands, services, events, and people, thus unveiling their nexus of similarity. When choosing a new smartphone, for instance, individuals compare models or brands based on features, performance, design, price, or reputation. This comparative analysis aids in finding the best match for personal preferences or values, ensuring informed decisions.

Comparison plays a pivotal role in learning and creativity. When acquiring new knowledge, individuals often compare it with existing information, facilitating a deeper understanding and encouraging the discovery of new connections and patterns. This skill is evident, for example, in language learning, where learners compare the structure and vocabulary of different languages, accelerating the learning process.

Comparative thinking extends beyond the mundane, fostering creativity and innovation. When individuals compare unrelated objects or entities,

they may generate novel ideas or solutions that can pave the way for groundbreaking inventions and discoveries, enriching society as a whole.

In essence, the ability to compare is a fundamental and invaluable skill that individuals employ across various situations and contexts. From guiding everyday choices to stimulating creative breakthroughs, comparative thinking serves as a cornerstone in navigating the complexities of the modern world.

1.1 Context

The output of the process of unveiling nexus of similarity essentially consists of two components: a *similarity measure* that quantifies the degree of resemblance, and a *nexus explanation* that describes the shared attributes or relations [33]. For instance, given the inputs ⟨Dukhan⟩, ⟨Paris⟩ and ⟨Rome⟩, the similarity measure could indicate that ⟨Paris⟩ and ⟨Dukhan⟩ are "slightly" similar, while ⟨Paris⟩ and ⟨Rome⟩ are "very" similar; or alternatively, assign numerical scores to reflect the relative similarity. The nexus explanation could state that ⟨Paris⟩ and ⟨Dukhan⟩ are both "cities", while ⟨Paris⟩ and ⟨Rome⟩ are also "European capitals located by rivers"; and these statements could be formalized using logical formulas similar to those shown in Figure 1.3.

Similarly, one can find the nexus of similarity between tuples of entities with more than one element. For example, ⟨Tokyo, Tokyo Tower⟩ and ⟨Paris, Eiffel Tower⟩ are "moderately" similar as they are both "pairs of a capital and one of its famous monuments that is a metal tower" [17].

In sports, a coach may compare the performance of different players based on their statistics, skills and the reputation of the teams for which they have previously played. For example, looking at the tuples of entities ⟨Lionel Messi, Barcelona⟩ and ⟨Cristiano Ronaldo, Juventus⟩ we can state that they are "highly" similar, as both pairings represent "world-class forwards paired with top European football clubs in which they played".

In cinematic critique, the evaluation of movies often encompasses genre, plot, and ratings. For instance, let us take it into consideration the tuples of entities ⟨The Godfather, Crime Drama⟩ and ⟨The Departed, Crime Thriller⟩. These tuples can be deemed "fairly" similar, as both represent pairings of "Films that have garnered Academy Awards, helmed by directors of Italian descent, paired with their main theme being organized crime". In fact, the first film is a distinguished work directed by Francis Ford Coppola, while the second is an esteemed creation helmed by Martin Scorsese.

This might seem like a trivial exercise, but it becomes more challenging,

interesting and valuable when applied to complex entities such as products, services, behaviors or health conditions. For instance, in e-commerce, Amazon may recommend certain laptops to a user who is looking for high-performance devices with touch screen, webcam, and Windows 11.

In tourism, an agency may suggest another beautiful beach resort in the Caribbean to a couple who enjoyed their previous vacations in similar destinations.

In social networks, Facebook may display the most relevant ads by analyzing the user's interests and preferences based on their previous clicks.

In streaming services, Netflix may offer personalized recommendations based on the user's viewing history and ratings.

In medicine, doctors may need to explain why some patients respond better to a certain treatment than others.

Artificial intelligence (AI) is the field of computer science that aims to create machines or software that can perform tasks that normally require human intelligence. One of the motivations for developing AI is to help humans with various problems or challenges that they face in their daily lives. Therefore, if something is important for humans, it is also important for AI to be able to deal with it. The tasks discussed above informally have been addressed in various ways over the years. To give one of the possible examples, *semantic similarity* is a concept that has been studied for more than a century by researchers from various disciplines, who have proposed different methods to quantify the degree of resemblance between entities, using either descriptive ratings or numerical scores [33]. For instance, modern computing machines have achieved a level of sophistication that enables them to compute a reasonably high similarity score between $\langle\mathsf{Paris}\rangle$ and $\langle\mathsf{Rome}\rangle$, by considering various aspects such as both being "European cities", "locations by rivers", "capitals of states that were founding members of the European Economic Community", and so forth. Furthermore, some methods are also capable of distinguishing that $\langle\mathsf{Paris}\rangle$ and $\langle\mathsf{Eiffel\ Tower}\rangle$ are not very similar, despite their strong relatedness [17]. Lastly, by following the same logic, machines are also able to classify $\langle\mathsf{Rome}\rangle$ as a "capital" rather than a "state" or a "park", by comparing similarity scores between the given entity and some class labels. In the last two decades, motivated by "Google Sets" [21], a significant amount of academic and industrial efforts have been dedicated to providing solutions for expanding a given set of entities with similar ones. The most common tasks in this area are *entity set expansion* [63], *entity recommendation* [10], *tuples expansion* [27], or *entity suggestion* [91]. For example, using existing methods one can expand the set $\mathbf{U} = \{\langle\mathsf{Paris}\rangle, \langle\mathsf{Rome}\rangle\}$ and obtain, for instance, $\mathbf{U}' = \mathbf{U} \cup \{\langle\mathsf{Amsterdam}\rangle\}$;

then, one can reapply the process starting from $\mathbf{U}'$ to obtain, for instance, the expanded set $\mathbf{U}'' = \mathbf{U}' \cup \{\langle \mathsf{Brussels} \rangle, \langle \mathsf{Rio\ de\ Janeiro} \rangle, \langle \mathsf{Vienna} \rangle\}$. Indeed, all the elements of these sets share one or more of the aforementioned properties, namely "European cities", "locations by rivers", etc. Conventional methods mainly measure similarities within textual corpora or tabular data [88, 40]. In recent years, there has been an increasing trend in exploiting structured knowledge bases (KBs), often represented as knowledge graphs (KGs)[18, 51]. Indeed, 'the heterogeneity, semantic richness and large-scale nature of knowledge base make traditional methods less effective'[54]. The work done to date is commendable and reflects what has been a great collective effort to provide solutions that reflect the need to uncover the nexus of similarity. However, it is imperative to recognize that these efforts are far from the end point. As will become evident in the next section, critical questions remain unresolved, which pave the way for the motivations and objectives that underpin the essence of this doctoral thesis.

1.2 Motivation and Objectives

The primary objective of this thesis is to enhance existing approaches in three fundamental ways.

Firstly, we introduce a versatile logic-based framework, accompanied by a formal semantics, designed to characterize the nexus of similarity between entities or tuples of entities within any knowledge base. This framework aims to express the nexus of similarity in a manner that is formal, comprehensive, and concise. The resulting explanations are crafted to be intelligible to both humans and machines, ensuring readability and interpretability.

Secondly, building upon our notion of characterization, we extend the traditional concept of linear expansions (i.e., $\mathbf{U} \subset \mathbf{U}' \subset \mathbf{U}'' \subset \ldots$) by demonstrating that expansions can be inherently taxonomic. Here, $\mathbf{U}$ may possess expansions that are incomparable under subset inclusion, enriching the depth and complexity of the comparison process.

Lastly, we delve into critical computational aspects of our framework, conducting a meticulous comparison with related approaches from the existing literature. This comparative analysis serves to elucidate the strengths and nuances of our proposed methodology.

In essence, this thesis focuses on a meticulous exploration of these aspects through the introduction of a novel logic-based framework. This framework not only provides a formal semantic foundation for the nexus of similarity but also enhances our understanding of the intricate relation-

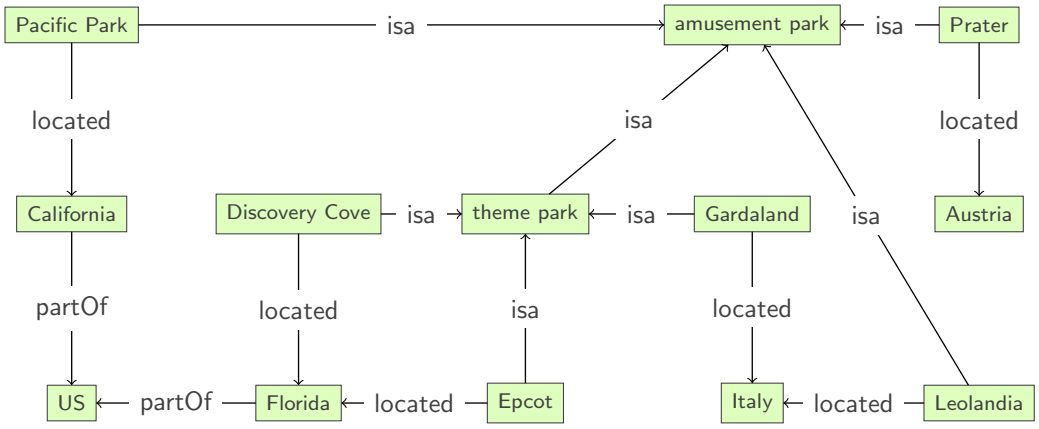

Figure 1.1: Knowledge graph $\mathcal{G}_0$ underlying the selective knowledge base $\mathcal{S}_0$ of Example 1.

ships between entities in various contexts.

To elucidate the fundamental principles underpinning our framework, we will illustrate its key concepts using a straightforward yet illuminating running example.

Example 1. *Let us assume that we have the KG $\mathcal{G}_0$ illustrated in Figure 1.1. This graph can be naturally encoded as the dataset:*

$$D_0 = \{\mathsf{isa}(\mathsf{Epcot}, \mathsf{tp}), \mathsf{located}(\mathsf{Epcot}, \mathsf{Florida}), \mathsf{partOf}(\mathsf{Florida}, \mathsf{US}), \ldots\}.$$

Here, tp *is a shorthand for* theme park *and* ap *for* amusement park*. Additionally, let us assume that we have an ontology O_0 consisting of one single Datalog rule:*

$$O_0 = \{\mathsf{isa}(x, z) \leftarrow \mathsf{isa}(x, y), \mathsf{isa}(y, z)\}.$$

The atoms entailed by the KB $K_0 = (D_0, O_0)$ are those in the set:

$$ent(K_0) = D_0 \ \cup \ \{\mathsf{isa}(c_1, c_3) : \mathsf{isa}(c_1, c_2) \in D_0 \wedge \mathsf{isa}(c_2, c_3) \in D_0\}.$$

Consider now the set $\mathbf{U}_0 = \{\langle \mathsf{Discovery\ Cove} \rangle, \langle \mathsf{Epcot} \rangle\}$, which can be referred to as an anonymous relation *or simply as a* unit*. To comprehensively express the nexus of similarity between the elements of $\mathbf{U}_0$ and unveil its expansions, it is indeed necessary to first establish a consensus on the relevant features or predicates describing any entity in D_0. Since such features or predicates might vary depending on the specific application scenario, we introduce the notion of* summary selector*. Intuitively, this selector is an algorithm that, for each entity e in D_0, chooses a subset of*

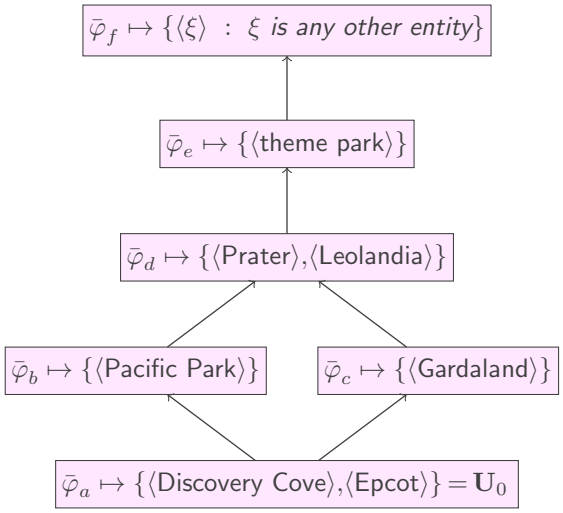

Figure 1.2: Expansion graph $eg(\mathbf{U}_0, \mathcal{S}_0)$. All formulas from $\tilde{\varphi_a}$ to $\tilde{\varphi_f}$ are expressed below in Figure 1.3.

$ent(K_0) \cup \{\top(e) : e$ is an entity in $K_0\}$, which is referred to as the summary *of e in the given scenario. For the purposes of our example, let us adopt the simple yet effective selector ς_0 that builds, for each entity e in D_0, the dataset $\varsigma_0(\langle e\rangle)$ as the union of the following sets:*

$$\begin{array}{rcl} A(e) & = & \{p(e', e'') \in ent(K_0) : e' = e\}, \\ B(e) & = & \{p'(e', e'') \in ent(K_0) : p(e, e') \in A(e) \wedge p \neq \mathsf{isa} \wedge p' \neq \mathsf{isa}\}, \\ C(e) & = & \{\top(e') : e' \text{ is an entity in } A(e) \cup B(e)\} \cup \{\top(e)\}. \end{array}$$

*For instance, the summary of $\langle$*Epcot*$\rangle$ is the dataset:*

$$\begin{array}{rcl} \varsigma_0(\langle\mathsf{Epcot}\rangle) & = & \{\mathsf{isa}(\mathsf{Epcot}, \mathsf{tp}), \mathsf{isa}(\mathsf{Epcot}, \mathsf{ap}), \mathsf{located}(\mathsf{Epcot}, \mathsf{Florida})\} \\ & & \cup \ \{\mathsf{partOf}(\mathsf{Florida}, \mathsf{US})\} \ \cup \\ & & \{\top(\mathsf{Epcot}), \top(\mathsf{tp}), \top(\mathsf{ap}), \top(\mathsf{Florida}), \top(\mathsf{US})\}, \end{array}$$

*while the summary of $\langle$*Discovery Cove*$\rangle$ is the dataset:*

$$\begin{array}{rcl} \varsigma_0(\langle\mathsf{Discovery\ Cove}\rangle) & = & \{\mathsf{isa}(\mathsf{Discovery\ Cove}, \mathsf{tp}), \mathsf{isa}(\mathsf{Discovery\ Cove}, \mathsf{ap}), \\ & & \mathsf{located}(\mathsf{Discovery\ Cove}, \mathsf{Florida})\} \\ & & \cup \ \{\mathsf{partOf}(\mathsf{Florida}, \mathsf{US})\} \ \cup \ \{\top(\mathsf{Discovery\ Cove}), \\ & & \top(\mathsf{tp}), \top(\mathsf{ap}), \top(\mathsf{Florida}), \top(\mathsf{US})\}. \end{array}$$

In the next sections, we will refer to the pair $\mathcal{S}_0 = (K_0, \varsigma_0)$ as a selective knowledge base *or SKB for brevity. By examining the formula*

$$\bar{\varphi}_1 = x \leftarrow \mathsf{isa}(x, \mathsf{ap}), \mathsf{located}(x, y), \mathsf{partOf}(y, \mathsf{US})$$

$$\bar{\varphi}_f = x \leftarrow \top(x)$$

$$\bar{\varphi}_e = x \leftarrow \mathsf{isa}(x, \mathsf{ap}), \top(x), \top(\mathsf{ap})$$

$$\bar{\varphi}_d = x \leftarrow \mathsf{isa}(x, \mathsf{ap}), \mathsf{located}(x, y), \top(x), \top(y), \top(\mathsf{ap})$$

$$\bar{\varphi}_c = x \leftarrow \mathsf{isa}(x, \mathsf{tp}), \mathit{conj}(\bar{\varphi}_d), \top(\mathsf{tp})$$

$$\bar{\varphi}_b = x \leftarrow \mathsf{isa}(x, \mathsf{ap}), \mathsf{located}(x, y), \top(x), \top(y), \top(\mathsf{ap}), \mathsf{partOf}(y, \mathsf{US}), \top(\mathsf{US})$$

$$\bar{\varphi}_a = x \leftarrow \mathsf{isa}(x, \mathsf{tp}), \mathit{conj}(\bar{\varphi}_e), \mathsf{located}(x, \mathsf{Florida}), \mathsf{partOf}(\mathsf{Florida}, \mathsf{US}), \top(\mathsf{tp}), \top(\mathsf{Florida}), \top(\mathsf{US})$$

Figure 1.3: Extended version of the formulas in Figure 1.2. Here tp stands for theme park, ap for amusement park, and $\mathit{conj}(\varphi)$ for the conjunction of atoms of φ.

in relation to the considered summaries, it is evident that $\bar{\varphi}_1$ *explains some nexus of similarity between* $\langle$Discovery Cove$\rangle$ *and* $\langle$Epcot$\rangle$*. However,* $\bar{\varphi}_1$ *neglects the additional information that both entities are also located in Florida according to their summaries. As it will become evident later, the following formula*

$$\bar{\varphi}_a = x \leftarrow \mathsf{isa}(x, \mathsf{tp}), \mathsf{isa}(x, \mathsf{ap}), \mathsf{located}(x, \mathsf{Florida}), \mathsf{partOf}(\mathsf{Florida}, \mathsf{US}), \top(x), \top(\mathsf{ap}), \top(\mathsf{tp}), \top(\mathsf{Florida}), \top(\mathsf{US})$$

given in Figure 1.3 fully explains the nexus of similarity between the two entities. Hence, we can assert that $\bar{\varphi}_a$ characterizes *their nexus of similarity.*

The last step is to classify each entity e *of* $\mathcal{S}_0$ *in relation to* $\mathbf{U}_0$*, by characterizing each unit* $\mathbf{U}_0 \cup \{\langle e \rangle\}$*. This leads to the* expansion graph *of* $\mathbf{U}_0$ *with respect to* $\mathcal{S}_0$*, denoted by* $eg(\mathbf{U}_0, \mathcal{S}_0)$ *and depicted in Figure 1.2. Intuitively, each node* n_1 *labeled by* $\varphi_1 \mapsto \mathbf{U}_1$ *says that* φ_1 *characterizes* $\mathbf{U}_0 \cup \{\langle e \rangle\}$ *for each* $\langle e \rangle \in \mathbf{U}_1$*. If there is a path from* n_1 *to another node* n_2 *labeled by* $\varphi_2 \mapsto \mathbf{U}_2$*, it means that* φ_2 *characterizes the unit* $\mathbf{U}_0 \cup \mathbf{U}_1 \cup \mathbf{U}_2$ *as well. Thus, we can conclude, for instance, that the nexus of similarity that* $\langle$Gardaland$\rangle$ *has with* $\mathbf{U}_0$ *incorporate those that* $\langle$Leolandia$\rangle$ *has with* $\mathbf{U}_0$*, showing that* Gardaland *is more similar to the entities of* $\mathbf{U}_0$ *than* Leolandia *with respect to* $\mathcal{S}$*. Additionally, the nexus of similarity that* $\langle$Pacific Park$\rangle$ *has with* $\mathbf{U}_0$ *are incomparable to those that* $\langle$Gardaland$\rangle$ *has with* $\mathbf{U}_0$*. In simple terms,* $eg(\mathbf{U}_0, \mathcal{S}_0)$ *is the expected taxonomic expansion of* $\mathbf{U}_0$*.* ■

After offering an informal overview of the primary components and concepts within our framework, we are poised to explore the intricate technical

details and challenges it encompasses. This thesis is dedicated to a meticulous and exhaustive examination of explaining anonymous relations through selective knowledge bases. Our objective is to uncover and define the nexus of similarity inherent in these relations, offering a structured characterization of the shared traits among its tuples. Additionally, we endeavor to showcase the practicality and value of our framework by applying it to real-world scenarios, evaluating its performance and quality rigorously.

To accomplish these objectives, our thesis is structured into distinct chapters, each focusing on a specific facet or task within our framework. Below, we provide a concise summary of the content and contributions of each chapter, accompanied by a brief discussion of the underlying challenges.

1.3 Contributions and Challenges

The structure of this thesis is meticulously organized to delve into the complexities of our framework. However, it is worth noting that part of the work presented here, even if duly reformulated and expanded, can be found in the following three works, the general impact of which we will briefly discuss on the thesis as a whole [3, 4, 2]. In the document [3], the reader is presented with a thorough compilation of our preliminary work. This encompasses a detailed examination of computational complexity and references the nascent stage of the framework that we will expound upon subsequently. Initially, this framework was predicated on the utilization of databases as opposed to knowledge bases. It is noteworthy that the results obtained during those early stages have been adeptly transposed into our novel approach. The article [4], which has been accorded the honor of publication in the Information Science Journal, provides an expansive overview of our contemporary framework. However, it omits an in-depth discussion on complexity outcomes and experimental facets. Furthermore, [2] offers an enlightening and elementary exposition of the salient concepts within our framework, alongside a preliminary presentation of the concept of *canonical characterization*. The principal aim of this latter publication was to disseminate knowledge to a broad readership while maintaining a discourse that is accessible and devoid of technical jargon.

Subsequent chapters contribute significantly to the elucidation of anonymous relations by employing selective knowledge bases. Herein, we furnish a synopsis of the content and the pivotal contributions of each chapter.

Chapter 2: Preliminaries In this foundational chapter, we establish the groundwork by defining essential notation and concepts, including conjunc-

tive formulas, knowledge bases, and complexity classes. We review pertinent existing results and techniques that form the basis for our framework.

Chapter 3: The Framework Here, we unveil the core contribution of this thesis: a formal framework for characterizing nexus of similarity. We systematically motivate the problem, defining anonymous relations and their representation through conjunctive formulas. Our explanation language, rooted in first-order logic with existential quantifiers, is introduced, showcasing its concise and intuitive expressiveness. Additionally, we define selective knowledge bases, a crucial element for our framework, and elucidate their properties and construction methods.

Chapter 4: Nexus Characterizations This chapter delves into the pivotal concept of characterization within our framework. We explore various forms, including canonical and core characterizations, and establish their relationships. We present algorithms for computing the canonical characterization of a unit and transitioning to its core, shedding light on the computational intricacies involved.

Chapter 5: Taxonomic Expansions Expanding our framework, we introduce the innovative concept of taxonomic expansions. These expansions offer a novel perspective on objects sharing similar nexus of similarity akin to the initial unit. We present the expansion graph, a formal object representing possible expansion paths, diverging from the traditional linear expansions. We explore its properties and structure, showcasing its potential for generating informative expansions for anonymous relations.

Chapter 6: Computational Aspects This chapter tackles critical computational challenges within our framework. Problems such as determining nexus of similarity between tuples of entities, finding core characterizations, and navigating the expansion graph are analyzed. We delve into the complexity and tractability of these tasks, offering efficient algorithms and delineating their tractability frontiers.

Chapter 7: A Prototypical Tool Here, we provide insights into the implementation aspects of our framework. We discuss design choices and optimization techniques employed in our prototype system. Experimental results from an ad hoc dataset, which is to be considered as a secondary contribution of the thesis, are presented, evaluating the performance and quality of our system across diverse anonymous relations.

Chapter 8: Related Work This chapter explores related works in the literature, spanning various directions like explanation generation, query rewriting, similarity measure construction, and data access based on ontology and description logics. We compare and contrast our framework with these approaches, addressing definability problems and complexities arising from our design choices.

Chapter 9: Discussion and Conclusions In the final chapter, we summarize the main contributions and findings of this work. We reflect on limitations and challenges, suggesting promising avenues for future research.

This thesis encompasses a comprehensive exploration of our framework, offering valuable insights and paving the way for further advancements in the field.

Chapter 2

Preliminaries

In this chapter, our primary focus is on establishing a solid foundation of notation and referencing well-known results essential for our study. By providing precise references and ensuring clarity in notation, we aim to familiarize the reader with the fundamental concepts and definitions crucial to our exploration. This foundational understanding will be instrumental in comprehending the complexities discussed in subsequent chapters.

2.1 Mathematical Background

In the upcoming sections, we will assume that the reader is acquainted with the fundamental concepts of *sets* [42], which represent collections of distinct objects, and *mappings* between sets [42], which establish rules assigning each element of one set to a unique element of another set. Additionally, familiarity with general set operations such as *union* (the set of all elements belonging to either set) and *intersection* (the set of all elements common to both sets) is presumed.

However, for the sake of clarity and to eliminate the need for constant back-and-forth referencing, we will provide brief definitions of mathematical tools when they become essential for our demonstrations or constructions. This approach ensures that our exposition remains clear, concise, rigorous, and complete.

This practice stems from our intention to present complex concepts and tools in an understandable manner. Some well-known tools, like the *direct product* [77], have been redefined to enhance explanatory convenience and simplify demonstrations. Despite these modifications, the essential properties enabling these tools to function as valuable instruments in our demonstrations have been retained.

In contrast, when referring to familiar objects such as sets or functions, we adhere to classical representations. Standard notation and terminology will always be used for these objects, unless stated otherwise.

Moreover, we assume a foundational understanding of statistical concepts, including *random variables* [34] (variables dependent on random experiment outcomes), *random samples* [34] (collections of independent and identically distributed random variables), *mean* [34] (a measure of central tendency), *variance* [34] (a measure of distribution spread), *distributions* [34] (mathematical models describing random variable behaviors), and *percentiles* [34] (values dividing a distribution equally). These concepts will be employed to describe and analyze data properties and algorithms. Additionally, specific distributions and statistical tests relevant to our study will be introduced.

2.2 Relational Structures

We are given three pairwise disjoint countably infinite lexicographically ordered sets of symbols: P of *predicates*, C of *constants*, and V of *variables*. When referring to variables, we typically employ symbols such as x, y, z, and variations thereof. Constants and variables are called *terms*. Each predicate p has an *arity* consisting of a positive integer denoted by $|p|$. The set P contains the special unary predicate $\top$ called *top*.

An *atom* is an expression of the form $p(t_1, \ldots, t_n)$, where p is a predicate of arity n and $t_1, \ldots, t_n$ is a *sequence* of terms. A *structure* is any set of atoms. The *domain* of a structure S is the set $\mathbb{D}_S$ of terms forming the atoms of S. A structure S is said to be *closed under* $\top$ if, for each $t \in \mathbb{D}_S$, it holds that $\top(t) \in S$. For a structure S, its *closure under* $\top$ is the structure $S^\top = S \cup \{\top(t) : t \in \mathbb{D}_S\}$. Therefore, a structure S is closed under $\top$ if, and only if, $S = S^\top$.

Example 2. *Consider the following structures:* $S_1 = \{\top(\mathsf{1})\}$, $S_2 = \{p(\mathsf{1}, \mathsf{2}), \top(\mathsf{1}), \top(\mathsf{2})\}$, $S_3 = \{s(\mathsf{1}), \top(\mathsf{2})\}$, *and* $S_4 = \{p_1(\mathsf{1}, x, \mathsf{1}), p_2(\mathsf{d}_1, \mathsf{1}, \mathsf{d}_2, \mathsf{c})\}$. *Accordingly, the domains of these four structures are:* $\mathbb{D}_{S_1} = \{\mathsf{1}\}$, $\mathbb{D}_{S_2} = \{\mathsf{1}, \mathsf{2}\}$, $\mathbb{D}_{S_3} = \{\mathsf{1}, \mathsf{2}\}$, *and* $\mathbb{D}_{S_4} = \{\mathsf{1}, \mathsf{c}, \mathsf{d}_1, \mathsf{d}_2, x\}$, *respectively. Hence,* S_1 *and* S_2 *are closed under* $\top$, *whereas* S_3 *and* S_4 *are not. Therefore,* $S_1^\top = S_1$ *and* $S_2^\top = S_2$; *on the contrary,* $S_3^\top = \{p(\mathsf{1}), \top(\mathsf{2}), \top(\mathsf{1})\} \neq S_3$ *and* $S_4^\top = \{p_1(\mathsf{1}, x, \mathsf{1}), p_2(\mathsf{d}_1, \mathsf{1}, \mathsf{d}_2, \mathsf{c}), \top(\mathsf{1}), \top(\mathsf{c}), \top(\mathsf{d}_1), \top(\mathsf{d}_2), \top(x)\} \neq S_4$. ■

A structure S is said to be *connected* if (i) $|S| = 1$ or (ii) there exist two connected structures S' and S'' such that both $S = S' \cup S''$ and $\mathbb{D}_{S'} \cap \mathbb{D}_{S''} \neq \emptyset$ hold. Note that condition (ii) of this inductive definition is

equivalent to requiring that (ii') there exist two connected structures $S' \subset S$ and $S'' \subset S$ such that $S = S' \cup S''$, $S' \cap S'' = \emptyset$ (namely, S' and S'' form a proper partition of S), and $\mathbb{D}_{S'} \cap \mathbb{D}_{S''} \neq \emptyset$. Note that, whenever S can be partitioned in two nonempty connected structures S' and S'' such that $\mathbb{D}_{S'} \cap \mathbb{D}_{S''} = \emptyset$, then S is not connected.

Example 3. *Let $S_1 = \{p(1)\}$, $S_2 = \{p(1), r(1, x)\}$, and $S_3 = \{p(1), r(2, x)\}$ be three structures. Therefore, S_1 is connected, as $|S_1| = 1$; S_2 is also connected, as $S_2 = \{p(1)\} \cup \{r(1, x)\}\}$ is the union of two structures of cardinality 1 such that the intersection of their domains is $\{1\}$; finally, S_3 is not connected, as the two structures $S' = \{p(1)\}$ and $S'' = \{r(2, x)\}$ form a partition of S and $\mathbb{D}_{S'} \cap \mathbb{D}_{S''} = \{1\} \cap \{2, x\} = \emptyset$.* ∎

Observe that the notion of connected structure can be equivalently given in terms of the *structure hypergraph* [14] or the *Gaifman graph* [11]. Indeed, for a structure S, its structure hypergraph $\mathcal{H}_S$ is the pair $(\mathbb{D}_S, H)$, where $H = \{\{t_1, \ldots, t_n\} : p(t_1, \ldots, t_n) \in S\}$; analogously, its Gaifman graph $\mathcal{G}_S$ is the pair $(\mathbb{D}_S, E)$, where $E = \{(t_i, t_j), (t_j, t_i) : p(t_1, \ldots, t_n) \in S \wedge 1 \leq i \leq j \leq n\}$. Accordingly, S is connected if, and only if, $\mathcal{H}_S$ is connected if, and only if, $\mathcal{G}_S$ is connected.

Definition 1. *Consider two structures S and S' together with a set T of terms. A T-*homomorphism *from S to S' is a total map $h : \mathbb{D}_S \to \mathbb{D}_{S'}$ such that both the next hold:*

(i) $h(t) = t$, for each $t \in T \cap \mathbb{D}_S$; and

(ii) $p(t_1, \ldots, t_n) \in S$ implies $p(h(t_1), \ldots, h(t_n)) \in S'$.

If $T = \emptyset$, then the prefix "T-" can be omitted. In what follows, a T-homomorphisms h can be explicitly given as a set of the form $\{t \mapsto h(t) \mid t \in \mathbb{D}_S\}$; moreover, given an atom $\alpha = p(t_1, \ldots, t_n)$, we will use $h(\alpha)$ as a shorthand for $p(h(t_1), \ldots, h(t_n))$. ∎

Before proceeding further, let us illustrate these notions via a simple example.

Example 4. *Let $S = \{p(\mathsf{a}, y), p(y, z), p(z, \mathsf{b})\}$ and $S' = \{p(\mathsf{a}, \mathsf{b}), p(\mathsf{b}, \mathsf{a}), r(\mathsf{b}, \mathsf{d})\}$ be two structures, and let $T = \mathsf{C}$. For instance, according to Definition 1, $h_1 = \{\mathsf{a} \mapsto \mathsf{a}, y \mapsto \mathsf{b}, z \mapsto \mathsf{a}, \mathsf{b} \mapsto \mathsf{b}\}$ is a C-homomorphism from S to S', while $h_2 = \{\mathsf{a} \mapsto \mathsf{a}, y \mapsto \mathsf{a}, z \mapsto \mathsf{b}, \mathsf{b} \mapsto \mathsf{b}\}$ and $h_3 = \{\mathsf{a} \mapsto \mathsf{b}, y \mapsto \mathsf{b}, z \mapsto \mathsf{a}, \mathsf{b} \mapsto \mathsf{a}\}$ are not. Indeed, while h_1 satisfies both conditions coming from Definition 1, concerning h_2, we have that $p(z, \mathsf{b}) \in S$,*

but $p(h_2(z), h_2(\mathsf{b})) = p(\mathsf{b}, \mathsf{b}) \notin S'$*, violating condition* (ii) *of Definition 1; and concerning* h_3*, although condition* (ii) *of Definition 1 holds, condition* (i) *does not. Indeed,* $\mathsf{a} \mapsto \mathsf{b}$ *and* $\mathsf{b} \mapsto \mathsf{a}$ *are two clear violations, since* $T \cap \mathbb{D}_S = \{\mathsf{a}, \mathsf{b}\}$*.* ■

Consider two structures S and S' admitting a T-homomorphism h from S to S'. If h is bijective, then it is called *T-isomorphism*, denoted by $S \simeq_T S'$. A *T-core* of a structure S is any $S' \subseteq S$ such that there is a T-homomorphism from S to S' and there is no $S'' \subset S'$ that admits a T-homomorphism from S' to S''. It is well-known that the T-core of a structure S is unique up to isomorphism, namely if S' and S'' are T-cores of S, then $S' \simeq_T S''$ [28]. Hence, it is common to talk about *the* T-core of a structure.

Example 5. *Consider the structure* $S = \{p(x, y), p(z, y), r(y, a)\}$*. It is easy to see that* S *contains two structures, namely* $S_1 = \{p(x, y), r(y, a)\}$ *and* $S_2 = \{p(z, y), r(y, a)\}$*, which both represent the core of* S *up to isomorphism. Indeed,* $S_1 \simeq_{\mathsf{C}} S_2$ *via the bijective* C*-homomorphism* $h = \{x \mapsto z, y \mapsto y, a \mapsto a\}$*. Also,* $h_1 = \{x \mapsto x, z \mapsto x, y \mapsto y, a \mapsto a\}$ *is a* C*-homomorphism from* S *to* S_1 *and* $h_2 = \{x \mapsto z, z \mapsto z, y \mapsto y, a \mapsto a\}$ *is a* C*-homomorphism from* S *to* S_2*. Finally, by removing some atom from* S_1 *or* S_2*, no homomorphism from* S *can exist.* ■

An n-ary *tuple* is an expression of the form $\langle t_1, \ldots, t_n \rangle$, where each t_i is a term. Intuitively, a tuple can be thought as an "anonymous" atom or, conversely, an atom can be thought as a "labeled" tuple. For a tuple τ (resp., sequence $\mathbf{s}$) of terms, its i-th term is denoted by $\tau[i]$ (resp., $\mathbf{s}[i]$). For a set T of terms, T^n denotes the set of all the $|T|^n$ tuples of arity n that can be constructed by using terms from T. As for atoms, the *domain* of a set J of tuples is the set $\mathbb{D}_J$ of terms forming the tuples of J. For example, the domain of $\{\langle \mathsf{1}, x, \mathsf{1} \rangle, \langle \mathsf{d}_1, \mathsf{d}_2, \mathsf{c} \rangle\}$ is the set $\{\mathsf{1}, \mathsf{c}, \mathsf{d}_1, \mathsf{d}_2, x\}$.

2.3 Knowledge Bases

From an informal perspective, a *knowledge base* represents a sophisticated amalgamation of two fundamental elements: an *extensional database* [24], like a *relational database* [36], containing factual data, and an *ontology* [73], serving as a structured framework encoding domain-specific knowledge in a logical format. This harmonious fusion of data and conceptual understanding enables the knowledge base to surpass the limitations of a conventional database.

In essence, we can envision these two components as a structure, as defined previously, accompanied by a set of rules capable of deriving new knowledge.

Formally, we can define a *dataset* as a finite nonempty structure D such that $\mathbb{D}_D \subset \mathsf{C}$. Subsequently, a *knowledge base* (KB) is a pair $K = (D, O)$, where D is a dataset and O is a (onto)logical theory expressed in some fragment of first-order logic [7, 15]. Conventionally, an atom α is *entailed* by K if it occurs in every model of K. The set of atoms entailed by K is denoted as $ent(K)$.

To further enhance the reader's understanding of this concept, we provide the following expample:

Example 6. *Consider the following dataset* $D = \{\mathsf{r}(\mathsf{a},\ \mathsf{b})\}$ *and the ontology* O *built starting from the single rule* $\mathsf{r}(\mathsf{x},\ \mathsf{y}) :- \mathsf{r}(\mathsf{y},\ \mathsf{x})$*, expressed as a classic Datalog rule. The pair* $K = (D,\ O)$ *is a knowledge base and* $ent(K)$ *is given by the set* $\{\mathsf{r}(\mathsf{a},\ \mathsf{b}), \mathsf{r}(\mathsf{b},\ \mathsf{a})\}$. ∎

2.4 Logic Formulas

In the forthcoming section, we will succinctly address the syntax and semantics of conjunctive formulas pertinent to subsets of first-order logic. Additionally, we will present some foundational results that will be instrumental throughout the entirety of this work.

Syntax

A *(conjunctive) formula*, CF for short, is an expression φ of the form

$$x_1, \ldots, x_n \leftarrow p_1(\mathbf{t}_1), \ldots, p_m(\mathbf{t}_m), \tag{2.1}$$

where $n \geq 0$ is its *arity*, $m > 0$ is its *size* often denoted by $|\varphi|$, each $\mathbf{t}_i$ is a sequence of terms, each $p_i(\mathbf{t}_i)$ is an atom, and each x_j is a variable —called *free*— occurring in some of the atoms of φ. Hereinafter, the sequence or conjunction $p_1(\mathbf{t}_1), \ldots, p_m(\mathbf{t}_m)$ of atoms is denoted by $conj(\varphi)$ and the set of its atoms by $atm(\varphi)$. Essentially, φ is a *primitive-positive formula* [69], PPF for short, without equality but equipped with constants.

A formula is said to be *open* if it contains at least one free variable, that is if $n \geq 1$. A formula φ is *connected* if $atm(\varphi)$ is a connected structure.

Semantics

Consider an n-ary CF φ. The *output* to φ over a dataset D is the set $\varphi(D)$ of every tuple $\langle t_1, \ldots, t_n \rangle$ that admits a C-homomorphism h from $atm(\varphi)$ to D, such that each $h(x_i) = t_i$. In the following section, we provide a detailed example designed to solidify the reader's understanding of the concepts we have recently introduced, particularly those pertaining to the syntax and semantics. This illustrative example will serve as a practical application of the theoretical elements discussed, ensuring a deeper comprehension that will be beneficial throughout the subsequent chapters of this work.

Example 7. *Consider again the dataset* $D = \{\mathsf{r}(\mathsf{a},\ \mathsf{b})\}$. *Now define the two CF formulas* φ_1 *and* φ_2 *as follows:*

$$\begin{aligned} \varphi_1 = x \quad &\leftarrow \quad \mathsf{r}(x,\ y) \\ \varphi_2 = x \quad &\leftarrow \quad \mathsf{r}(x,\ x). \end{aligned}$$

Although both are valid formulas from a syntactic point of view, the second, in relation to the dataset represented, has no confirmation, as there is no element belonging to the domain of the structure D *present simultaneously in the first and second positions in an atom in* D. *However, it is clear that* $\varphi_1(D) = \{\langle \mathsf{a} \rangle\}$. ∎

Syntax and semantics are not sufficient for our purposes, it is in fact necessary to talk about the relationships between formulas, or rather when one formula is more specific than another or if perhaps they are incomparable with each other.

Relationships between Formulas

Consider two formulas φ_1 and φ_2 with free variables $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$, respectively. Hereinafter, we will use the notation $\varphi_1 \longrightarrow \varphi_2$ to indicate the existence of a C-homomorphism h from $atm(\varphi_1)$ to $atm(\varphi_2)$ such that $y_i = h(x_i)$, for each $i = 1, \ldots, n$.

Example 8. *Consider the following formulas*

$$\begin{aligned} \bar{\varphi}_a = x \quad &\leftarrow \quad \mathsf{isa}(x, \mathsf{tp}), \mathsf{isa}(x, \mathsf{ap}), \mathsf{located}(x, \mathsf{Florida}), \mathsf{partOf}(\mathsf{Florida}, \mathsf{US}), \\ &\qquad \top(x), \top(\mathsf{ap}), \top(\mathsf{tp}), \top(\mathsf{Florida}), \top(\mathsf{US}) \\ \bar{\varphi}_b = x \quad &\leftarrow \quad \mathsf{isa}(x, \mathsf{ap}), \mathsf{located}(x, y), \top(x), \top(y), \top(\mathsf{ap}), \mathsf{partOf}(y, \mathsf{US}), \\ &\qquad \top(\mathsf{US}) \end{aligned}$$

given in Figure 1.2. We have that $\bar{\varphi}_b \longrightarrow \bar{\varphi}_a$. *Indeed, we can consider the* C*-homomorphism* h *from* $atm(\bar{\varphi}_b)$ *to* $atm(\bar{\varphi}_a)$ *where* $h = \{t \mapsto t \mid t \in \mathbb{D}_{atm(\bar{\varphi}_b)} \setminus \{y\}\} \cup \{y \mapsto \mathsf{Florida}\}$. *Moreover, one can also verify that, for each formula* φ *in Figure 1.2, it holds that* $\varphi \longrightarrow \bar{\varphi}_a$. ■

If both $\varphi_1 \longrightarrow \varphi_2$ and $\varphi_2 \not\longrightarrow \varphi_1$ (i.e., $\varphi_2 \longrightarrow \varphi_1$ does not hold), then we say that φ_1 is a *generalization of* φ_2. Moreover, if both $\varphi_1 \longrightarrow \varphi_2$ and $\varphi_2 \longrightarrow \varphi_1$ do not hold, then the two formulas are said *incomparable*.

Example 9. *Consider again the formulas* $\bar{\varphi}_a$ *and* $\bar{\varphi}_b$ *given in the previous example. We have seen that* $\bar{\varphi}_b \longrightarrow \bar{\varphi}_a$. *Moreover, since* $\mathsf{Florida} \in \mathbb{D}_{atm(\bar{\varphi}_a)}$, *but* $\mathsf{Florida} \notin \mathbb{D}_{atm(\bar{\varphi}_b)}$, *there cannot exist any* C*-homomorphism from* $atm(\bar{\varphi}_a)$ *to* $atm(\bar{\varphi}_b)$. *Hence,* $\bar{\varphi}_b$ *is a generalization of* $\bar{\varphi}_a$. *Moreover, by considering any formula* φ *in Figure 1.2 different from* $\bar{\varphi}_a$, *then* φ *is a generalization of* $\bar{\varphi}_a$. *Now, consider the formula*

$$\bar{\varphi}_c = x \leftarrow \mathsf{isa}(x, \mathsf{ap}), \mathsf{located}(x, y), \top(x), \top(y), \top(\mathsf{ap}), \mathsf{isa}(x, \mathsf{tp}), \top(\mathsf{tp})$$

also given in Figure 1.2. We can observe that $\bar{\varphi}_b$ *and* $\bar{\varphi}_c$ *are incomparable. Indeed, since* $\mathsf{US} \in \mathbb{D}_{atm(\bar{\varphi}_b)}$, *but* $\mathsf{US} \notin \mathbb{D}_{atm(\bar{\varphi}_c)}$, *there cannot exist any* C*-homomorphism from* $atm(\bar{\varphi}_b)$ *to* $atm(\bar{\varphi}_c)$. *Conversely, since* $\mathsf{tp} \in \mathbb{D}_{atm(\bar{\varphi}_c)}$, *but* $\mathsf{tp} \notin \mathbb{D}_{atm(\bar{\varphi}_b)}$, *there cannot exist any* C*-homomorphism from* $atm(\bar{\varphi}_c)$ *to* $atm(\bar{\varphi}_b)$. ■

Additionally, if $\varphi_1 \longleftrightarrow \varphi_2$ (i.e., both $\varphi_1 \longrightarrow \varphi_2$ and $\varphi_2 \longrightarrow \varphi_1$ hold), then the two formulas are said *equivalent*. We close the section with some useful properties related to this last notion. For completeness, we report also a brief and simple proof of the well known results.

Proposition 1. *Let* φ_1 *and* φ_2 *be two formulas. Then,* $\varphi_1 \longrightarrow \varphi_2$ *implies* $\varphi_2(D) \subseteq \varphi_1(D)$, *for each dataset* D.

Proof. Denote by $x_1, \ldots, x_n$ the free variables of φ_1 and by $y_1, \ldots, y_n$ the free variables of φ_2, in this exact order. Since $\varphi_1 \longrightarrow \varphi_2$ holds by hypothesis, we have a C-homomorphism h from $atm(\varphi_1)$ to $atm(\varphi_2)$, such that $h(x_i) = y_i$ for each $i \in \{1, \ldots, n\}$. Let now $\tau = \langle t_1, \ldots, t_n \rangle$ be a tuple in $\varphi_2(D)$. We want to show that $\tau \in \varphi_1(D)$, that is, there exists a C-homomorphism h_1 from $atm(\varphi_1)$ to D such that $h_1(x_1), \ldots, h_1(x_n) = t_1, \ldots, t_n$. To this aim, notice that since $\tau \in \varphi_2(D)$, then by definition of output of a formula over a dataset, there is a C-homomorphism h_2 from $atm(\varphi_2)$ to D such that $h_2(y_1), \ldots, h_2(y_n) = t_1, \ldots, t_n$. Therefore, $h_1 = h_2 \circ h$ is a C-homomorphism from $atm(\varphi_1)$ to D such that $h_1(x_1), \ldots, h_1(x_n) = t_1, \ldots, t_n$. Hence, τ also belongs to $\varphi_1(D)$. Therefore, $\varphi_2(D) \subseteq \varphi_1(D)$. □

From the previous Proposition 1, we can derive the following corollary.

Corollary 1. *Let φ_1 and φ_2 be two formulas. Then, $\varphi_1 \longleftrightarrow \varphi_2$ implies $\varphi_1(D) = \varphi_2(D)$, for each dataset D.*

Proof. Assume that $\varphi_1 \longleftrightarrow \varphi_2$. Hence, $\varphi_1 \longrightarrow \varphi_2$ and $\varphi_2 \longrightarrow \varphi_1$ do hold. By applying Proposition 1 to $\varphi_1 \longrightarrow \varphi_2$, we have that $\varphi_2(D) \subseteq \varphi_1(D)$, for each dataset D, whereas by applying Proposition 1 to $\varphi_2 \longrightarrow \varphi_1$, we have that $\varphi_1(D) \subseteq \varphi_2(D)$, for each dataset D. Therefore, we can conclude that $\varphi_1(D) = \varphi_2(D)$, for each dataset D. □

2.5 Computational Complexity

This section does not aim to replace a complete digression on the theory of complexity and computability; instead, it intends for the reader to familiarize themselves with the notation used in this work.

From here on, we assume, for brevity, that the reader is familiar with the concept of *Turing Machines*, both deterministic and non-deterministic, as models of computation.

The section does not delve deeply into the idea of what a *problem* really is. Instead, a problem, or a *task*, for us is nothing more than a formal object consisting of an input, a question about it, and an output. Often, the output of the task will be of the "yes" or "no" type, and in this specific case, following [64], we refer to them as decision problems/tasks.

For these decision tasks, we consider *classes* such as $\mathsf{NL} \subseteq \mathsf{P} \subseteq \mathsf{NP}$ and $\mathsf{coNP} \subseteq \mathsf{DP} \subseteq \mathsf{PH} \subseteq \mathsf{PSPACE} = \mathsf{NPSPACE} \subseteq \mathsf{EXP} \subseteq \mathsf{NEXP}$, where DP (resp., DEXP) is the closure of $\mathsf{NP} \cup \mathsf{coNP}$ (resp., $\mathsf{NEXP} \cup \mathsf{coNEXP}$) under intersection.

We assume that the reader is familiar with complexity classes in general, but we will provide some examples of problems belonging to the less frequent classes.

Specifically, conventional instances within the complexity classes DP and DEXP often consist of paired problems that permit independent questioning. However, as we will elucidate in subsequent chapters, the scope of these classes extends beyond such isolated instances. The most famous example of DP problems is $\mathsf{SAT} - \mathsf{UNSAT}$, an input of which typically consists of a pair of formulas for which we ask the satisfiability of one and the unsatisfiability of the other respectively. Further details can be found in Theorem 17.1 of Papadimitriou's book [64].

For an example concerning a problem belonging to the exponential counterpart of this computational class, namely DEXP, one can think of a generic

pair of instances $(\mathsf{P}_1, \mathsf{P}_2)$ where P_1 belongs to NEXP and P_2 belongs to coNEXP.

Of course, since DP (resp., DEXP) is the closure of $\mathsf{NP} \cup \mathsf{coNP}$ (resp., $\mathsf{NEXP} \cup \mathsf{coNEXP}$) under intersection, we also know that any problem in NP and coNP (resp., in NEXP and coNEXP) also belongs to DP (resp., DEXP). However, these examples do not provide real insights into the class itself and could be misleading; hence, they were not chosen as the first example.

Not all problems are suited for decision tasks, depending on various factors. In some cases, instead of merely knowing if a truth assignment for a formula φ exists, we would like to actually "see" this truth assignment. Problems of this kind, where we desire to effectively "return" part of our computation rather than a simple binary answer, are referred to as *functional problems* or *functional tasks*.

In this thesis, readers will come across the following classes of functional tasks: $F\mathsf{P} = F\mathsf{P}^{\mathsf{NL}} \subseteq F\mathsf{P}^{\mathsf{NP}} \subseteq F\mathsf{P}^{\mathsf{PH}} \subset F\mathsf{PSPACE} = F\mathsf{PSPACE}^{\mathsf{NPSPACE}} \subseteq F\mathsf{EXP}^{\mathsf{NP}}$, where $F\mathsf{PSPACE}$ represents problems solvable by a Turing Machine with a one-way, write-only unlimited output tape and constantly many work tapes of polynomial length.

Additionally, we assume that the reader is familiar with the concept of *reduction*, a technique allowing the comparison of different decision problems by transforming one problem into another. For detailed information on this topic, readers are referred to Papadimitriou's book [64].

Chapter 3

The Framework

In this chapter, we systematically present the core components of our formal framework. The discourse commences with the introduction of the critical concept of *anonymous relation*. The foundation of our framework is predicated on the following principle: given a generic *unit* of knowledge, namely an *anonymous relation*, in conjunction with pertinent information—sourced from a *summary* provided by a *summary selector* within a *selective knowledge base*—our objective is to construct a thorough *explanation*. For this purpose, we will utilize a specially devised language. The aforementioned trio—comprising the *anonymous relation*, the *selective knowledge base*, and the *explanation*—are the foundational pillars of our formal framework, each playing an integral role in its structure and function.

3.1 Anonymous Relations

Definition 2. *An n-ary* anonymous relation *is any finite nonempty set of n-ary tuples of constants, where $n > 0$. An anonymous relation will often be referred to as a* unit*.* ■

Intuitively, in fact, an anonymous relation encodes some "unit" of knowledge that we would like to characterize and possibly expand. We use the term anonymous relation because in fact this object behaves like relations, i.e. a subset of the Cartesian product, for which however, unlike the classical case, we do not have any relation name. For instance, one may consider the unary anonymous relation $\mathbf{U}_0 = \{\langle \mathsf{Discovery\_Cove} \rangle, \langle \mathsf{Epcot} \rangle\}$ in Example 1.

3.2 Selective Knowledge Bases

We now introduce the notion of *summary selector* to collect relevant features or predicates describing a tuple τ with respect to a KB K, aligning with the requirements of the scenario at hand.

Definition 3. *A* summary selector *is a computable function ς that:*

(*i*) *takes as input a KB $K = (D, O)$ plus an n-ary tuple τ of constants from $\mathbb{D}_D$; and*

(*ii*) *returns a dataset $\varsigma(K, \tau) \subseteq ent(K)^\top$ still closed under $\top$ —representing a* summary *of τ (w.r.t. K)— whose domain contains at least all the constants of τ.*

If there is no ambiguity, we write $\varsigma(\tau)$ instead of $\varsigma(K, \tau)$. ■

From a practical perspective, ς can be thought as an algorithm taking in input the pair (K, τ) and returning the dataset $\varsigma(K, \tau)$.

Example 10. *According to Example 1, consider the tuple $\tau = \langle \mathsf{Discovery\_Cove} \rangle$ with respect to the KB K_0 together with three computable functions ς_1, ς_2, and ς_3, that behave as follows:*

$$\begin{aligned}
\varsigma_1(K_0, \tau) &= \{\mathsf{located}(\mathsf{Discovery_Cove}, \mathsf{Florida}), \top(\mathsf{Discovery_Cove}), \\
&\quad \top(\mathsf{Florida}), \top(\mathsf{Epcot})\} \\
\varsigma_2(K_0, \tau) &= \{\mathsf{located}(\mathsf{Discovery_Cove}, \mathsf{Florida})\} \\
\varsigma_3(K_0, \tau) &= \{\mathsf{located}(\mathsf{Epcot}, \mathsf{Florida}), \top(\mathsf{Epcot}), \top(\mathsf{Florida})\}
\end{aligned}$$

According to Definition 3, since the dataset $\varsigma_1(K_0, \tau)$ is closed under $\top$ and since it contains Discovery_Cove*, then it is a summary of τ; differently, both $\varsigma_2(K_0, \tau)$ and $\varsigma_3(K_0, \tau)$ are not. Indeed, $\varsigma_2(K_0, \tau)$ is not closed under $\top$, whereas $\varsigma_3(K_0, \tau)$ does not contain* Discovery_Cove*, which occurs in τ.* ■

Let us now discuss the specific summary selector used in Example 1.

Example 11. *Consider the specific computable function ς_0 used in our running example, where, for each entity e in D_0, $\varsigma_0(K_0, \langle e \rangle)$ is the union of the following three sets:*

$$\begin{aligned}
&A(e) = \{p(e', e'') \in ent(K_0) : e' = e\} \\
&B(e) = \{p'(e', e'') \in ent(K_0) : p(e, e') \in A(e) \wedge p \neq \mathsf{isa} \wedge p' \neq \mathsf{isa}\} \\
&C(e) = \{\top(e') : e' \text{ is an entity in } A(e) \cup B(e)\} \cup \{\top(e)\}.
\end{aligned}$$

It is not difficult to see that ς_0 satisfies Definition 3 and thus it is a summary selector. Indeed, by construction, $\varsigma(K_0, \langle e \rangle) \subseteq ent(K_0)^\top$, and moreover $C(e) \subseteq \varsigma_0(K_0, \langle e \rangle)$ implies that $\varsigma_0(K_0, \langle e \rangle)$ is closed under $\top$ and its domain contains the constant e. For instance, when $e = \mathsf{Discovery\_Cove}$, we have that

$$\begin{aligned} A(e) &= \{\mathsf{isa}(\mathsf{Discovery\_Cove}, \mathsf{tp}), \mathsf{isa}(\mathsf{Discovery\_Cove}, \mathsf{ap}), \\ &\quad \mathsf{located}(\mathsf{Discovery\_Cove}, \mathsf{Florida})\} \\ B(e) &= \{\mathsf{partOf}(\mathsf{Florida}, \mathsf{US})\} \\ C(e) &= \{\top(\mathsf{Discovery\_Cove}), \top(\mathsf{tp}), \top(\mathsf{ap}), \top(\mathsf{Florida}), \top(\mathsf{US})\} \end{aligned}$$

Intuitively, in $A(e)$ we select all knowledge directly connected (at distance 1) to our entity; in $B(e)$ we select all knowledge connected at distance 2 with our entity, but without involving the relation isa*; and in $C(e)$ we select all the necessary top atoms.* ■

In addition to the simple selector ς_0 in Example 1, in case K is an RDF graph and τ is a unary tuple, the notions of *concise bounded description*[1] and *rooted RDF-graph* [23] can be seen as possible summaries of τ with respect to K. Recently, novel techniques that work with unary tuples [52] and binary tuples [68] have been also proposed, offering further options and approaches for summary selection. For a discussion on the actual usefulness of summaries, we refer the reader to the Section 9.1.

Definition 4. *Let K be a knowledge base, and let ς be a summary selector. Then, a* selective knowledge base*, SKB for short, is the pair $\mathcal{S} = (K, \varsigma)$.* ■

For an example of selective knowledge base, we refer the reader to the SKB $\mathcal{S}_0$ given in Example 1. Note, however, that SKBs can encode and accommodate different kind of existing KBs, such as those based on Knowledge Graphs (KGs), RDF Graphs, Description Logics (DLs), Existential Rules, and so on. For instance, in Example 1, one could encode the initial KG as the following dataset: $D'_0 = \{\mathsf{triple}(\mathsf{Epcot}, \mathsf{isa}, \mathsf{tp}), \mathsf{triple}(\mathsf{Epcot}, \mathsf{located}, \mathsf{Florida}), \mathsf{triple}(\mathsf{Florida}, \mathsf{partOf}, \mathsf{US}), \ldots\}$, where labeled arcs in Figure 1.1 are encoded via ternary atoms over the predicate triple, as usual in RDF. However, we can even deal with arities of predicates that go beyond 2 (the maximum in KGs and DLs) and 3 (the maximum in RDF), as common in Database Theory and in several fields of Knowledge Representation and Reasoning.

[1] See https://www.w3.org/Submission/2005/SUBM-CBD-20050603/

3.3 Explanation Language

We have reached the third fundamental block of our formal framework, namely the explanation language. As we have already done previously, we will begin by dividing the section into two parts, one for the syntax and one for the semantics linked to the language. In the ensuing discourse, we will articulate some fundamental results concerning the language that is central to the thesis at hand. It is imperative to note that a detailed exposition of the design choices is reserved for Chapter 9. This sequencing is intentional and stems from the necessity of having all elements of the framework meticulously aligned before such discussions could be meaningful. The arduous task of ensuring each component was precisely positioned could not have been feasibly accomplished earlier in the thesis. As such, the subsequent chapters are designed to methodically set each piece in its rightful place, thereby laying a solid foundation upon which the rationale behind our design choices can be thoroughly understood.

Syntax

We inaugurate this section with a definition that is essential for a profound comprehension of the subsequent material.

Definition 5. *Consider an open formula φ as in Equation 2.1. An atom $\alpha \in atm(\varphi)$ is connected to some free variable of φ if there exists a connected structure $S \subseteq atm(\varphi)$ such that both $\alpha \in S$ and $\mathbb{D}_S \cap \{x_1, \ldots, x_n\} \neq \emptyset$ hold. Accordingly, we say that φ is* nearly connected *if each of its atoms is connected to some free variable of φ.* ■

In a sense, an open formula φ is nearly connected if the structure $S = atm(\varphi) \cup \{\mathsf{free}(x_1, \ldots, x_n)\}$ is connected, where $\mathsf{free}(x_1, \ldots, x_n)$ is a dummy fresh atom. Intuitively, in the latter case, each $\alpha \in atm(\varphi)$ has a (direct or indirect) connection with some atom containing at least one free variable. Clearly, if an open formula is connected, then it is also nearly connected; however, the converse may not hold.

Example 12. *For instance, formula $\bar{\varphi}_1 = x \leftarrow \mathsf{isa}(x, \mathsf{ap}), \mathsf{located}(x, y), \mathsf{partOf}(y, \mathsf{US})$ —stating that "x is an amusement park, located in y, that is part of US"— is (nearly) connected. Indeed, $atm(\bar{\varphi}_1) = \{\mathsf{isa}(x, \mathsf{ap}), \mathsf{located}(x, y), \mathsf{partOf}(y, \mathsf{US})\}$ is a connected structure.*

Whereas, formula $\bar{\varphi}_2 = x, y \leftarrow \mathsf{isa}(x, \mathsf{ap}), \mathsf{partOf}(y, \mathsf{US})$ —stating that "the pair (x, y) is such that x is an amusement park and y is part of US"— is nearly connected but not connected. Indeed, $atm(\bar{\varphi}_2) = \{\mathsf{isa}(x, \mathsf{ap}),$

$\mathsf{partOf}(y, \mathsf{US})\}$ *is not connected, since* $\mathbb{D}_{\{\mathsf{isa}(x,\mathsf{ap})\}} \cap \mathbb{D}_{\{\mathsf{partOf}(y,\mathsf{US})\}} = \{x, \mathsf{ap}\} \cap \{y, \mathsf{US}\} = \emptyset$ *and there is no other way to divide* $atm(\bar{\varphi}_2)$ *into two connected structures; on the other hand,* $atm(\bar{\varphi}_2) \cup \{\mathsf{free}(x, y)\} = \{\mathsf{isa}(x, \mathsf{ap}), \mathsf{partOf}(y, \mathsf{US}), \mathsf{free}(x, y)\}$ *is a connected structure, hence,* $\bar{\varphi}_2$ *is a nearly connected formula.*

Differently, formula $\bar{\varphi}_3 = x \leftarrow \mathsf{isa}(x, \mathsf{ap}), \mathsf{partOf}(y, \mathsf{US})$*— stating that "x is an amusement park and there exists a place y that is part of US"— is not even nearly connected as* $atm(\bar{\varphi}_3) \cup \{\mathsf{free}(x)\} = \{\mathsf{isa}(x, \mathsf{ap}), \mathsf{partOf}(y, \mathsf{US}), \mathsf{free}(x)\}$ *is not a connected structure.* ■

Consider an n-ary open formula φ. Let ϱ_φ be the bijection than maps the j-th free variable of φ to z_j and each non-free variable y of φ to y_φ. Let φ' be the formula obtained from φ by replacing each variable t with $\varrho_\varphi(t)$. The *conjunction* of two n-ary open formulas φ_1 and φ_2, denoted by $\varphi_1 \wedge \varphi_2$, is the new formula $z_1, \ldots, z_n \leftarrow conj(\varphi'_1), conj(\varphi'_2)$.

Example 13. *Let* φ_1 *be the binary formula* $x_1, x_2 \leftarrow p(a, x_1), r(x_2, y)$ *and let* φ_2 *be the binary formula* $y_2, y_1 \leftarrow u(y_1), v(y_2, b, y, a)$*. Then, we obtain that* φ'_1 *is the formula* $z_1, z_2 \leftarrow p(a, z_1), r(z_2, y_{\varphi_1})$ *and* φ'_2 *is* $z_1, z_2 \leftarrow u(z_2), v(z_1, b, y_{\varphi_2}, a)$*. Consequently, the conjunction* $\varphi_1 \wedge \varphi_2$ *is the new formula* $z_1, z_2 \leftarrow p(a, z_1), r(z_2, y_{\varphi_1}), u(z_2), v(z_1, b, y_{\varphi_2}, a)$*. Note that,* φ_1 *and* φ_2 *are nearly connected formulas, and so is also their conjunction* $\varphi_1 \wedge \varphi_2$*. As we will see shortly, this is no mere coincidence.* ■

A set F of formulas is *closed under conjunction* if, for each φ_1 and φ_2 in F, also $\varphi_1 \wedge \varphi_2$ is in F. In the subsequent sections, we utilize *nearly connected conjunctive formulas*, N$\mathbb{C}$F for short, as the formalism to explain the nexus of similarity between tuples within a unit, while considering their summaries. The rationale behind the choice of this language is discussed in Section 9.1. We conclude with a rather intuitive yet useful result of our language.

Proposition 2. N$\mathbb{C}$F *is closed under conjunction.*

Proof. We have to prove that if two n-ary formulas φ_1 and φ_2 are in N$\mathbb{C}$F, then $\varphi_3 = z_1, \ldots, z_n \leftarrow conj(\varphi'_1), conj(\varphi'_2)$ is in N$\mathbb{C}$F too, where φ'_i is the formula obtained from φ_i by replacing each variable t with $\varrho_{\varphi_i}(t)$, for each $i \in \{1, 2\}$ and ϱ_{φ_i} is the bijection than maps the j-th free variable of φ_i to z_j and each non-free variable y of φ_i to y_{φ_i}. To this aim, according to Definition 5, we need to show (i) that φ_3 is an open formula and (ii) that $atm(\varphi_3) \cup \{\mathsf{free}(z_1, \ldots, z_n)\}$ is a connected structure. To show (i) it suffice to recall that, by hypotheses, both φ_1 and φ_2 are two n-ary formulas in N$\mathbb{C}$F. In particular, since, by Definition 5, formulas in N$\mathbb{C}$F are open, φ_3

is an open formula. The last statement holds since, again by definition, the conjunction of two n-ary open formulas is an n-ary open formula. In order to show (ii) we need to find two connected structure, call them S_1 and S_2 respectively, such that (a) $\mathbb{D}_{S_1} \cap \mathbb{D}_{S_2} \neq \emptyset$ and (b) $atm(\varphi_3) \cup \{\mathsf{free}(z_1, \ldots, z_n)\}$ is equal to $S_1 \cup S_2$. To this aim, let $S_1 = atm(\varphi_1') \cup \{\mathsf{free}(z_1, \ldots, z_n)\}$ and $S_2 = atm(\varphi_2') \cup \{\mathsf{free}(z_1, \ldots, z_n)\}$. We now need to prove (a), (b) and also that both S_1 and S_2 are connected. First notice that since φ_1 and φ_2 are two n-ary open formulas and both ϱ_{φ_1} and ϱ_{φ_2} are, by their definition, two bijections that preserve the number of free variables of a formula we get that φ_1' and φ_2' are in N$\mathbb{C}$F and $z_1, \ldots, z_n$ are their free variables. Now, since φ_1' and φ_2' are in N$\mathbb{C}$F, the fact that both S_1 and S_2 are connected follows from Definition 5. (a) follows from $\{\mathsf{free}(z_1, \ldots, z_n)\} \subseteq S_1 \cap S_2$ together with the fact that $n > 0$, since both φ_1' and φ_2' are in N$\mathbb{C}$F. (b) follows from the fact that by construction $atm(\varphi_3) = atm(\varphi_1') \cup atm(\varphi_2')$, hence $atm(\varphi_3) \cup \{\mathsf{free}(z_1, \ldots, z_n)\} = atm(\varphi_1') \cup atm(\varphi_2') \cup \{\mathsf{free}(z_1, \ldots, z_n)\} = atm(\varphi_1') \cup \{\mathsf{free}(z_1, \ldots, z_n)\} \cup atm(\varphi_2') \cup \{\mathsf{free}(z_1, \ldots, z_n)\} = S_1 \cup S_2$. □

Semantics

An *instance* of φ according to some selective knowledge base $\mathcal{S} = (K, \varsigma)$ is any tuple $\tau \in \varphi(\varsigma(\tau))$. Intuitively, if τ is not in the output to φ over its summary $\varsigma(\tau)$, then φ does not express properties of τ in terms of the considered scenario; if so, we consider τ not an instance of φ according to $\mathcal{S}$. The set of all φ-instances is denoted by $inst(\varphi, \mathcal{S})$. It is important to note the following:

Proposition 3. *Consider some selective KB $\mathcal{S} = (D, \varsigma)$ and some formula φ. It holds that $inst(\varphi, \mathcal{S}) \subseteq \varphi(ent(K)^\top)$.*

Proof. Let φ be an n-ary formula as in Equation 2.1 and let $\tau = \langle t_1, \ldots, t_n \rangle$ be a tuple in $inst(\varphi, \mathcal{S})$. We have to show $\tau \in \varphi(ent(K)^\top)$, in other words we need to find a C-homomorphism h from $atm(\varphi)$ to $ent(K)^\top$, such that $h(x_1), \ldots, h(x_n) = t_1, \ldots, t_n$. To this aim, first notice that given a dataset D' such that $D' \subseteq ent(K)^\top$, then any C-homomorphism from $atm(\varphi)$ to D' is also a C-homomorphism from $atm(\varphi)$ to $ent(K)^\top$. Hence, to show $\tau \in \varphi(ent(K)^\top)$, it suffice to (i) find a dataset $D' \subseteq ent(K)^\top$ and (ii) materialize a C-homomorphism, call it h', from $atm(\varphi)$ to D' such that $h'(x_1), \ldots, h'(x_n) = t_1, \ldots, t_n$. For (i), since, by Definition 3, we know that $\varsigma(\tau)$ is a dataset and $\varsigma(\tau) \subseteq ent(K)^\top$, we can choose $D' = \varsigma(\tau)$. For (ii), since by hypotheses $\tau \in inst(\varphi, \mathcal{S})$, the C-homomorphism h' from $atm(\varphi)$ to D' such that $h'(x_1), \ldots, h'(x_n) = t_1, \ldots, t_n$ exists by hypotheses. □

Let us now illustrate the new notion with the aid of two examples. The first one shows that the inclusion $inst(\varphi, \mathcal{S}) \subseteq \varphi(ent(K)^\top)$ given in the statement of Proposition 3 may be strict in some cases.

Example 14. *Consider now the formula*

$$\bar{\varphi}_4 = y \leftarrow \mathsf{located}(x, y),\ \mathsf{partOf}(y, \mathsf{US})$$

together with the KB $K_0 = (D_0, O_0)$ in Example 1. We have that $\bar{\varphi}_4(ent(K_0)^\top) = \{\langle \mathsf{Florida} \rangle, \langle \mathsf{California} \rangle\}$; indeed the following maps

$$\begin{array}{rcl} h_1 & = & \{x \mapsto \mathsf{Epcot}, y \mapsto \mathsf{Florida}, \mathsf{US} \mapsto \mathsf{US}\} \\ h_2 & = & \{x \mapsto \mathsf{Discovery\_Cove}, y \mapsto \mathsf{Florida}, \mathsf{US} \mapsto \mathsf{US}\} \\ h_3 & = & \{x \mapsto \mathsf{Pacific\_Park}, y \mapsto \mathsf{California}, \mathsf{US} \mapsto \mathsf{US}\} \end{array}$$

are all and only the C*-homomorphisms from $atm(\bar{\varphi}_4)$ to $ent(K_0)^\top$. In particular, $h_1(y) = h_2(y) = \mathsf{Florida}$ and $h_3(y) = \mathsf{California}$. Moreover, we also have that $inst(\bar{\varphi}_4, \mathcal{S}_0) = \emptyset$. This is because, for each $c \in \{\mathsf{Florida}, \mathsf{California}\}$, it holds that $\varsigma_0(\langle c \rangle) = \{partOf(c, \mathsf{US}), \top(c), \top(\mathsf{US})\}$, implying that $\bar{\varphi}_4(\varsigma_0(\langle c \rangle)) = \emptyset$.* ■

The following example shows that the inclusion $inst(\varphi, \mathcal{S}) \subseteq \varphi(ent(K)^\top)$ given in the statement of Proposition 3 may be satisfied with equality in some cases.

Example 15. *Consider the formula*

$$\bar{\varphi}_5 = x \leftarrow \mathsf{located}(x, y),\ \mathsf{partOf}(y, \mathsf{US})$$

and the KB K_0 in Example 1. Note that $\bar{\varphi}_5$ differs from $\bar{\varphi}_4$ of Example 14 only in the free variable. We have that $\bar{\varphi}_5(ent(K_0)^\top) = \{\langle \mathsf{Epcot} \rangle, \langle \mathsf{Discovery\_Cove} \rangle, \langle \mathsf{Pacific\_Park} \rangle\}$; indeed $h_1 = \{x \mapsto \mathsf{Epcot}, y \mapsto \mathsf{Florida}, \mathsf{US} \mapsto \mathsf{US}\}$, $h_2 = \{x \mapsto \mathsf{Discovery\_Cove}, y \mapsto \mathsf{Florida}, \mathsf{US} \mapsto \mathsf{US}\}$, and $h_3 = \{x \mapsto \mathsf{Pacific\_Park}, y \mapsto \mathsf{California}, \mathsf{US} \mapsto \mathsf{US}\}$ are all and only the three C*-homomorphisms from $atm(\bar{\varphi}_5)$ to D_0. In particular, $h_1(x) = \mathsf{Epcot}$, $h_2(x) = \mathsf{Discovery\_Cove}$ and $h_3(x) = \mathsf{Pacific\_Park}$. Moreover, $inst(\bar{\varphi}_5, \mathcal{S}_0) = \bar{\varphi}_5(D_0)$. Indeed, we have to consider the following summaries*

$$
\begin{aligned}
\varsigma_0(\langle \mathsf{Epcot} \rangle) &= \{\mathsf{located}(\mathsf{Epcot}, \mathsf{Florida}), \mathsf{partOf}(\mathsf{Florida}, \mathsf{US}), \\
&\quad \mathsf{isa}(\mathsf{Epcot}, \mathsf{tp}), \mathsf{isa}(\mathsf{Epcot}, \mathsf{ap}), \top(\mathsf{Epcot}), \\
&\quad \top(\mathsf{Florida}), \top(\mathsf{US}), \top(\mathsf{tp}), \top(\mathsf{ap})\}; \\
\varsigma_0(\langle \mathsf{Discovery\_Cove} \rangle) &= \{\mathsf{located}(\mathsf{Discovery\_Cove}, \mathsf{Florida}), \top(\mathsf{tp}), \top(\mathsf{ap}), \\
&\quad \mathsf{isa}(\mathsf{Discovery\_Cove}, \mathsf{tp}), \mathsf{isa}(\mathsf{Discovery\_Cove}, \mathsf{ap}), \\
&\quad \mathsf{partOf}(\mathsf{Florida}, \mathsf{US}), \top(\mathsf{Discovery\_Cove}), \\
&\quad \top(\mathsf{US}), \top(\mathsf{Florida})\}; \\
\varsigma_0(\langle \mathsf{Pacific\_Park} \rangle) &= \{\mathsf{located}(\mathsf{Pacific\_Park}, \mathsf{California}), \top(\mathsf{Pacific\_Park}), \\
&\quad \mathsf{isa}(\mathsf{Pacific\_Park}, \mathsf{ap}), \mathsf{partOf}(\mathsf{California}, \mathsf{US}), \\
&\quad \top(\mathsf{California}), \top(\mathsf{US}), \top(\mathsf{ap})\}.
\end{aligned}
$$

Therefore, h_1, h_2, and h_3 are also C*-homomorphisms, respectively, from $atm(\bar{\varphi_5})$ to $\varsigma_0(\langle \mathsf{Epcot} \rangle)$, from $atm(\bar{\varphi_5})$ to $\varsigma_0(\langle \mathsf{Discovery\_Cove} \rangle)$, and from $atm(\bar{\varphi_5})$ to $\varsigma_0(\langle \mathsf{Pacific\_Park} \rangle)$ such that the following three conditions apply $h_1(x) = \mathsf{Epcot}$, $h_2(x) = \mathsf{Discovery\_Cove}$, and $h_3(x) = \mathsf{Pacific\_Park}$.* ∎

As it should be clear, the introduction of the instances of formula is triggered by the use of summaries. However, one may wonder whether this notion is really appropriate or whether the output of a formula is also a good choice. To give an ultimate answer to these questions, we refer the reader to the Section 9.1. We close the section with some useful properties needed in order to prove other results throughout the thesis.

Proposition 4. *Consider a selective KB $\mathcal{S} = (D, \varsigma)$ and two formulas φ_1 and φ_2 of the same arity. It holds that $inst(\varphi_1 \wedge \varphi_2, \mathcal{S}) = inst(\varphi_1, \mathcal{S}) \cap inst(\varphi_2, \mathcal{S})$.*

Proof. We will divide the proof into two parts, first we will show (i) that $inst(\varphi_1 \wedge \varphi_2, \mathcal{S}) \subseteq inst(\varphi_1, \mathcal{S}) \cap inst(\varphi_2, \mathcal{S})$ and then we will show (ii) that $inst(\varphi_1 \wedge \varphi_2, \mathcal{S}) \supseteq inst(\varphi_1, \mathcal{S}) \cap inst(\varphi_2, \mathcal{S})$. This is meaningful since, by Proposition 2, $\varphi_1 \wedge \varphi_2 \in \mathrm{N}\mathbb{C}\mathrm{F}$. Let us start by showing point (i). To this aim, let $\varphi_3 = \varphi_1 \wedge \varphi_2$, and without losing in generality let $x_1, \ldots, x_n$ be the free variables of φ_1 and $y_1, \ldots, y_n$ be the free variables of φ_2, both considered in this exact order. By definition, φ_3 is the formula $z_1, \ldots, z_n \leftarrow conj(\varphi_1'), conj(\varphi_2')$, where φ_i' is the formula obtained from φ_i by replacing each variable t with $\varrho_{\varphi_i}(t)$, for each $i \in \{1, 2\}$ and ϱ_{φ_i} is the bijection than maps the j-th free variable of φ_i to z_j and each non-free variable y of φ_i to y_{φ_i}. Consider now an n-ary tuple τ of the form $\langle t_1, \ldots, t_n \rangle$ such that $\tau \in inst(\varphi_3, \mathcal{S})$. What we have to show is that $\tau \in inst(\varphi_1, \mathcal{S}) \cap inst(\varphi_2, \mathcal{S})$. In other words we need to find two C-homomorphisms h_1 and h_2, respectively from $atm(\varphi_1)$ to $\varsigma(\tau)$ and from $atm(\varphi_2)$ to $\varsigma(\tau)$, such that

$h_1(x_1), \ldots, h_1(x_n) = h_2(y_1), \ldots, h_2(y_n) = t_1, \ldots, t_n$. To this aim, first notice that since $\tau \in inst(\varphi_3, \mathcal{S})$, by definition we know that there exists a C-homomorphism h from $atm(\varphi_3)$ to $\varsigma(\tau)$ such that $h(z_1), \ldots, h(z_n) = t_1, \ldots, t_n$. Now consider the following two C-homomorphism, $h'_1 = \{t \mapsto h(t) \ : \ t \in \mathbb{D}_{atm(\varphi'_1)}\}$ from $atm(\varphi'_1)$ to $\varsigma(\tau)$ and $h'_2 = \{t \mapsto h(t) \ : \ t \in \mathbb{D}_{atm(\varphi'_2)}\}$ from $atm(\varphi'_2)$ to $\varsigma(\tau)$. By construction we have that the sequence $h'_1(z_1), \ldots, h'_1(z_n)$ is equal to $h'_2(z_1), \ldots, h'_2(z_n)$, that is equal to $h(z_1), \ldots, h(z_n) = t_1, \ldots, t_n$. We can now consider the following two C-homomorphism, $h_1 = h'_1 \circ \varrho_{\varphi_1}$ from $atm(\varphi_1)$ to $\varsigma(\tau)$ and $h_2 = h'_2 \circ \varrho_{\varphi_2}$ from $atm(\varphi_2)$ to $\varsigma(\tau)$. Again by construction $h_1(x_1), \ldots, h_1(x_n) = h_2(y_1), \ldots, h_2(y_n) = h(z_1) \ , \ldots, h(z_n) = t_1, \ \ldots, \ t_n$, hence both $\tau \in inst(\varphi_1, \mathcal{S})$ and $\tau \in inst(\varphi_2, \mathcal{S})$ do hold, therefore $\tau \in inst(\varphi_1, \mathcal{S}) \cap inst(\varphi_2, \mathcal{S})$. This completes the first part of the proof. Now we need to show point (ii). To this aim consider again a tuple τ of the form $\langle t_1, \ldots, t_n \rangle$ such that $\tau \in inst(\varphi_1, \mathcal{S}) \cap inst(\varphi_2, \mathcal{S})$. By definition, this means that two distinct C-homomorphism, h_1 from $atm(\varphi_1)$ to $\varsigma(\tau)$ and h_2 from $atm(\varphi_2)$ to $\varsigma(\tau)$ with $h_1(x_1), \ldots, h_1(x_n) = h_2(y_1), \ldots, h_2(y_n) = t_1, \ldots, t_n$ do exist. Consider now the two C-homomorphisms $h'_1 = h_1 \circ \varrho^{-1}_{\varphi_1}$ from $atm(\varphi'_1)$ to $\varsigma(\tau)$ and $h'_2 = h_2 \circ \varrho^{-1}_{\varphi_2}$ from $atm(\varphi'_2)$ to $\varsigma(\tau)$. By construction $h'_1(z_1), \ldots, h'_1(z_n) = h'_2(z_1), \ldots, h'_2(z_n) = t_1, \ldots, t_n$. From h'_1 and h'_2 we can construct a C-homomorphism h from $atm(\varphi_1 \wedge \varphi_2)$ to $\varsigma(\tau)$ such that $h(z_1), \ldots, h(z_n) = t_1, \ldots, t_n$. To this aim it suffices to consider $h = \{t \mapsto h'_1(t) \ : \ t \in \mathbb{D}_{atm(\varphi'_1)}\} \ \cup \ \{t \mapsto h'_2(t) \ : \ t \in \mathbb{D}_{atm(\varphi'_2)}\}$. Finally, we notice that h is a witness for $\tau \in inst(\varphi_1 \wedge \varphi_2, \mathcal{S})$, hence $inst(\varphi_1 \wedge \varphi_2, \mathcal{S}) \supseteq inst(\varphi_1, \mathcal{S}) \cap inst(\varphi_2, \mathcal{S})$ that is what we wanted to show. □

Another extremely useful property, as well as expected, since typically when working with conjunctive queries via the classic notion of answer (or output) this is the classic result that is obtained [16], is the following.

Proposition 5. *Let φ_1 and φ_2 be two formulas. Then, $\varphi_1 \longrightarrow \varphi_2$ implies $inst(\varphi_2, \mathcal{S}) \subseteq inst(\varphi_1, \mathcal{S})$, for each SKB $\mathcal{S}$.*

Proof. From Proposition 1 we know that, given these hypotheses, $\varphi_2(D) \subseteq \varphi_1(D)$. Now, we prove that $\varphi_2(D) \subseteq \varphi_1(D)$ for each D implies $inst(\varphi_2, \mathcal{S}) \subseteq inst(\varphi_1, \mathcal{S})$. To this aim, first notice that, by Definition 3, for each tuple τ and for each summary selector ς, we know that $\varsigma(\tau)$ is a dataset. Hence, if one knows that $\varphi_2(D) \subseteq \varphi_1(D)$ for each dataset D, one could derive $\varphi_2(\varsigma(\tau)) \subseteq \varphi_1(\varsigma(\tau))$. Then, $\tau \in \varphi_2(\varsigma(\tau))$ implies $\tau \in \varphi_1(\varsigma(\tau))$. Therefore, $inst(\varphi_2, \mathcal{S}) \subseteq inst(\varphi_1, \mathcal{S})$. □

We've reached a point where we can present the following corollary. This

is not just the result of our analysis; it is also the final key finding of this chapter.

Corollary 2. *Let φ_1 and φ_2 be two formulas. Then, $\varphi_1 \longleftrightarrow \varphi_2$ implies $inst(\varphi_1, \mathcal{S}) = inst(\varphi_2, \mathcal{S})$, for each SKB $\mathcal{S}$.*

Proof. Assume that $\varphi_1 \longleftrightarrow \varphi_2$. Hence, $\varphi_1 \longrightarrow \varphi_2$ and $\varphi_2 \longrightarrow \varphi_1$ do hold. By applying Proposition 5 to $\varphi_1 \longrightarrow \varphi_2$, we can easily derive that $inst(\varphi_2, \mathcal{S}) \subseteq inst(\varphi_1, \mathcal{S})$, for each SKB $\mathcal{S}$; whereas by applying Proposition 5 to $\varphi_2 \longrightarrow \varphi_1$, we have that$inst(\varphi_1, \mathcal{S}) \subseteq inst(\varphi_2, \mathcal{S})$, for each SKB $\mathcal{S}$. Therefore, we can conclude that $inst(\varphi_1, \mathcal{S}) = inst(\varphi_2, \mathcal{S})$, for each SKB $\mathcal{S}$. □

As previously stated, we refer the reader to Chapter 9 for further justifications on some of our design choices.

Chapter 4

Nexus Characterizations

In this Chapter, we aim to utilize nearly connected formulas and their semantics, as expressed through instances, to introduce the essential concepts of *explanations* and *characterizations* of a unit. We will establish that these objects always exist, firmly rooting them as fundamental elements of our discussion. Then, we show how to construct a *canonical* characterization of a unit. Finally, since canonical characterizations may be unnecessarily large, we show how to construct characterizations consisting of a minimal number of atoms, called *core* characterizations.

In what follows, consider to have an SKB $\mathcal{S} = (K, \varsigma)$ with $K = (D, O)$ together with an n-ary unit $\mathbf{U} = \{\tau_1, \ldots, \tau_m\}$ such that $\mathbb{D}_{\mathbf{U}} \subseteq \mathbb{D}_D$, and we refer $\mathbf{U}$ as an *$\mathcal{S}$-unit*.

4.1 Explanations

Intuitively, an explanation for $\mathbf{U}$ with respect to $\mathcal{S}$ is any nearly connected conjunctive formula that expresses some interconnected properties (shared by all elements) of $\mathbf{U}$.

Definition 6. *A formula φ* explains *some nexus of similarity between the tuples of the $\mathcal{S}$-unit $\mathbf{U}$ if both the following conditions hold: (i) φ is an* NCF*; and (ii) $inst(\varphi, \mathcal{S}) \supseteq \mathbf{U}$.*

Accordingly, we may also say that φ explains $\mathbf{U}$*, that φ* is an explanation for $\mathbf{U}$*, or that $\mathbf{U}$* is explained by φ *(with respect to $\mathcal{S}$).* ∎

We now illustrate this notion via a simple example.

Example 16. *Consider again our running example (Example 1), and the formula $\bar{\varphi}_1 = x \leftarrow \mathsf{isa}(x, \mathsf{ap}), \mathsf{located}(x, y), \mathsf{partOf}(y, \mathsf{US})$ of Example 12. By Example 12, we know that $\bar{\varphi}_1$ is an* NCF*. Moreover, $inst(\bar{\varphi}_1, \mathcal{S}_0) =$*

$\{\langle \mathsf{Discovery\_Cove}\rangle, \langle \mathsf{Epcot}\rangle, \langle \mathsf{Pacific\_Park}\rangle\} \supseteq \{\langle \mathsf{Epcot}\rangle, \langle \mathsf{Discovery\_Cove}\rangle\} = \mathbf{U}_0$. *Therefore,* $\bar{\varphi}_1$ *explains* $\mathbf{U}_0$. *Intuitively, this means that* Epcot *and* Discovery_Cove *are "amusement parks located somewhere within the United States".* ■

Since summaries are closed under $\top$, the next result follows.

Proposition 6. *The given* $\mathcal{S}$*-unit* $\mathbf{U}$ *always admits an explanation.*

Proof. Let φ be the formula $x_1, \ldots, x_n \leftarrow \top(x_1), \ldots, \top(x_n)$. We claim that φ explains $\mathbf{U}$. First, to get $\varphi \in \mathrm{N}\mathbb{C}\mathrm{F}$ one can observe that: (a) by Definition 2, arity of $\mathbf{U}$ is greater than zero, implying that φ is an open formula; and (b) $\{\top(x_1), \ldots, \top(x_n)\} \cup \{\mathsf{free}(x_1, \ldots, x_n)\}$ is connected, hence φ satisfies Definition 5. Second, to get $inst(\varphi, \mathcal{S}) \supseteq \mathbf{U}$, one can observe that, by Definition 3, both the following conditions hold: $\mathbb{D}_{\{\tau\}} \subseteq \mathbb{D}_{\varsigma(K,\tau)}$, for each $\tau \in \mathbf{U}$, and $\top(\tau[i]) \in \varsigma(K, \tau)$, for each $i \in \{1, \ldots, n\}$. □

One may note, however, that formula $\bar{\varphi}_1$ mentioned in Example 16 misses an important aspect. Intuitively, although $inst(\bar{\varphi}_1, \mathcal{S}_0) \supseteq \mathbf{U}_0$ holds, $\bar{\varphi}_1$ fails to account that the entities in $\mathbf{U}_0$ are "located in Florida". In a sense, $\bar{\varphi}_1$ only partially explains the nexus of similarity between the tuples of $\mathbf{U}_0$ with respect to $\mathcal{S}_0$. As an effect, $\langle \mathsf{Pacific\_Park}\rangle$ —which is not in Florida— also belongs to $inst(\bar{\varphi}_1, \mathcal{S}_0)$. Such shortcomings are overcome by the notion of characterization, introduced in the subsequent section.

4.2 Characterizations

Since arbitrary explanations may miss some nexus of similarity between the tuples of a unit, we introduce the following refined notion.

Definition 7. *A formula* φ characterizes *the nexus of similarity between the tuples of the given* $\mathcal{S}$*-unit* $\mathbf{U}$ *if both the next conditions hold:* (i) φ *explains* $\mathbf{U}$*; and* (ii) $\varphi \longrightarrow \varphi'$ *implies* $\varphi' \longrightarrow \varphi$ *whenever* φ' *explains* $\mathbf{U}$*. Accordingly, we may say, for short, that* φ *characterizes* $\mathbf{U}$*, that* φ *is a characterization for* $\mathbf{U}$*, or that* $\mathbf{U}$ *is characterized by* φ *(with respect to* $\mathcal{S}$*).* ■

Intuitively, φ is one of the least general explanation of $\mathbf{U}$ w.r.t. $\mathcal{S}$.

Example 17. *The formula* $\bar{\varphi}_1$ *of Example 16 explains the* $\mathcal{S}_0$*-unit* $\mathbf{U}_0$*, but does not characterize it with respect to* $\mathcal{S}_0$*. Indeed, by considering, for instance, the following formula*

$$\bar{\varphi}_a = x \;\leftarrow\; \mathsf{isa}(x, \mathsf{tp}), \mathit{conj}(\bar{\varphi}_e), \mathsf{located}(x, \mathsf{Florida}), \\ \mathsf{partOf}(\mathsf{Florida}, \mathsf{US}), \top(\mathsf{tp}), \top(\mathsf{Florida}), \top(\mathsf{US})$$

presented in Figure 1.2, we observe that $\bar{\varphi}_a$ *also explains* $\mathbf{U}_0$*. Moreover,* $\bar{\varphi}_1 \longrightarrow \bar{\varphi}_a$ *holds, but* $\bar{\varphi}_a \longrightarrow \bar{\varphi}_1$ *does not. Indeed,* $\bar{\varphi}_a$ *does characterize* $\mathbf{U}_0$*, and the reason will be clarified later in this section.* ■

Before formally proving that any unit always admits a characterization, we first show that whenever two characterizations exist, they are necessarily equivalent.

Proposition 7. *Any two characterizations for* $\mathbf{U}$ *are equivalent.*

Proof. Consider two explanations φ_1 and φ_2 both characterizing $\mathbf{U}$ and, by contradiction, suppose $\varphi_1 \longleftrightarrow \varphi_2$ does not hold. Consider now the formula $\varphi = \varphi_1 \wedge \varphi_2$ and remember that, by definition, φ is the formula $z_1, \ldots, z_n \leftarrow \mathit{conj}(\varphi_1'), \mathit{conj}(\varphi_2')$, where φ_i' is the formula obtained from φ_i by replacing each variable t with $\varrho_{\varphi_i}(t)$, for each $i \in \{1, 2\}$ and ϱ_{φ_i} is the bijection than maps the j-th free variable of φ_i to z_j and each non-free variable y of φ_i to y_{φ_i}. We will first show that $\varphi_2 \longrightarrow \varphi$ and $\varphi_1 \longrightarrow \varphi$ both hold and then get an absurd by C-homomorphism composition. Due to ϱ_{φ_1} and ϱ_{φ_2} we know that $\varphi_1 \longrightarrow \varphi$ and $\varphi_2 \longrightarrow \varphi$ both hold. By definition of characterization both φ_1 and φ_2 are explanations for $\mathbf{U}$. By Proposition 2, we know that $\varphi \in \mathrm{N}\mathbb{C}\mathrm{F}$ and, by Proposition 4, $\mathit{inst}(\varphi, \mathcal{S}) = \mathit{inst}(\varphi', \mathcal{S}) \cap \mathit{inst}(\varphi'', \mathcal{S}) \supseteq \mathbf{U}$, hence, since φ satisfies Definition 6, also φ explains $\mathbf{U}$. Since φ_1 characterize $\mathbf{U}$, $\varphi_1 \longrightarrow \varphi$ and φ is an explanation for $\mathbf{U}$, then, by Definition 7, $\varphi \longrightarrow \varphi_1$ holds and, since $\varphi_2 \longrightarrow \varphi$, by C-homomorphism composition we get $\varphi_2 \longrightarrow \varphi_1$. By a symmetrical argument also $\varphi_1 \longrightarrow \varphi_2$ holds. Therefore, we have shown that φ_1 and φ_2 are equivalent, reaching a contradiction. □

The aforementioned result allows us to reformulate Definition 7 in a more convenient way. This is done via the following corollary.

Corollary 3. *A formula* φ characterizes *the given* $\mathcal{S}$*-unit* $\mathbf{U}$ *if, and only if, both the following hold:* (*i*) φ *explains* $\mathbf{U}$*; and* (*ii*) $\varphi' \longrightarrow \varphi$ *whenever* φ' *explains* $\mathbf{U}$*.*

What remains to be determined is whether every unit admits such a formula. The following theorem provides a positive answer to this question.

Theorem 1. *The given* $\mathcal{S}$*-unit* $\mathbf{U}$ *always admits a characterization.*

The next section is devoted to prove this key result.

4.3 Canonical characterizations

We first recall and adapt to our purposes some well-known notions from database theory. Then, we show how to explicitly build a *canonical characterization* of the n-ary unit $\mathbf{U}=\{\tau_1,\ldots,\tau_m\}$ according to $\mathcal{S}=(K,\varsigma)$, called $can(\mathbf{U},\mathcal{S})$. Finally, since canonical characterizations always exist, we are able to derive Theorem 1.

Definition 8. *Consider the n-ary tuples $\bar{\tau}_1,\ldots,\bar{\tau}_\ell$. Their* direct product, *hereinafter denoted by $\bar{\tau}_1\otimes\ldots\otimes\bar{\tau}_\ell$, is the sequence $d_{\bar{\mathbf{s}}_\mathbf{1}},\ldots,d_{\bar{\mathbf{s}}_\mathbf{n}}$ of constants, where $\bar{\mathbf{s}}_\mathbf{i}$ is the sequence $\bar{\tau}_1[i],\ldots,\bar{\tau}_\ell[i]$, for each $i=1,\ldots,n$.*[1] *Accordingly, given k datasets $D_1,\ldots,D_k$, their* direct product *is the dataset*

$$\{p(\langle c_1^1,..,c_1^n\rangle\otimes..\otimes\langle c_k^1,..,c_k^n\rangle)\ :\ p(c_1^1,..,c_1^n)\in D_1,..,p(c_k^1,..,c_k^n)\in D_k\}$$

hereinafter denoted by $D_1\otimes\ldots\otimes D_k$. ■

Let us illustrate these notions with a simple example.

Example 18. *Consider the following three binary tuples $\langle 1,2\rangle$, $\langle 3,4\rangle$, and $\langle 5,6\rangle$. Hence, their direct product is the sequence*

$$\langle 1,2\rangle\otimes\langle 3,4\rangle\otimes\langle 5,6\rangle = d_{1,3,5},d_{2,4,6}.$$

Now, consider the following two datasets

$$D_1=\{p(\mathsf{a},\mathsf{c}),p(\mathsf{c},\mathsf{e}),p(\mathsf{e},\mathsf{b})\}\ \textit{and}\ D_2=\{p(\mathsf{a},\mathsf{b}),p(\mathsf{b},\mathsf{a}),r(\mathsf{b},\mathsf{d})\}.$$

Then, their direct product is the dataset

$$D_1\otimes D_2=\{p(d_{\mathsf{a},\mathsf{a}},d_{\mathsf{c},\mathsf{b}}),\ p(d_{\mathsf{a},\mathsf{b}},d_{\mathsf{c},\mathsf{a}}),\ p(d_{\mathsf{c},\mathsf{a}},d_{\mathsf{e},\mathsf{b}}),\ p(d_{\mathsf{c},\mathsf{b}},d_{\mathsf{e},\mathsf{a}}),\ p(d_{\mathsf{e},\mathsf{a}},d_{\mathsf{b},\mathsf{b}}),\ p(d_{\mathsf{e},\mathsf{b}},d_{\mathsf{b},\mathsf{a}})\}.$$

Note that, since there is no element in D_1 capable of being in direct product with the element $r(\mathsf{b},\mathsf{d})$ of D_2, the final structure is devoid of the predicate r. ■

The following results are easy consequences of our definitions.

[1] The direct product of tuples is usually defined as a binary operation. Given two n-ary tuples, $\tau=\langle\tau[1],\ldots,\tau[n]\rangle$ and $\tau'=\langle\tau'[1],\ldots,\tau'[n]\rangle$, then $\tau\otimes\tau'$ is the n-ary tuple $\langle\langle\tau[1],\tau'[1]\rangle,\ldots,\langle\tau[n],\tau'[n]\rangle\rangle$. Hence, this operation is associative up to isomorphisms, and it is possible to consider the direct product of more than two tuples [9, 77]. As we are not interested in a reiterate application of the direct product operator, for notational convenience, our direct product of two n-ary tuples is not an n-ary tuple, but just a sequence of fresh constants.

Proposition 8. *The direct product of datasets closed under $\top$ is a dataset closed under $\top$.*

Proof. Let $D_1, \ldots, D_k$ be datasets. By Definition 8, their direct product $D_1 \otimes \ldots \otimes D_k$ is the structure

$$P = \left\{ p(\langle c_1^1, .., c_1^n \rangle \otimes .. \otimes \langle c_k^1, .., c_k^n \rangle) \ : \ p(c_1^1, .., c_1^n) \in D_1, .., p(c_k^1, .., c_k^n) \in D_k \right\}.$$

What we have to show is that, for each $d_{c^1,\ldots,c^k} \in \mathbb{D}_P$, we have $\top(d_{c^1,\ldots,c^k}) \in P$. To this aim, let $d_{c^1,\ldots,c^k} \in \mathbb{D}_P$. By construction, $d_{c^1,\ldots,c^k} \in \mathbb{D}_P$ implies $c^1 \in \mathbb{D}_{D_1}, \ldots, c^k \in \mathbb{D}_{D_k}$. Since $D_1, \ldots, D_k$ are datasets, we know that $\top(c^1) \in D_1, \ldots, \top(c^k) \in D_k$. Hence, again by construction, $\top(d_{c^1,\ldots,c^k}) = \top(\langle c^1 \rangle \otimes \ldots \otimes \langle c^k \rangle) \in P$, that is what we wanted to show. □

From Proposition 8 one immediately derives the following:

Corollary 4. *The direct product of summaries is never empty.*

For a dataset D and an n-ary unit $\mathbf{U} = \{\tau_1, \ldots, \tau_m\}$, the direct product $P = D \otimes \ldots \otimes D$ —multiplying D with itself m times— has been already used in database theory to check whether $\mathbf{U}$ admits a CQ (i.e., constant-free CF) φ such that $\varphi(D) = \mathbf{U}$ [9]. In particular, let $d_{\mathbf{s}_1}, \ldots, d_{\mathbf{s}_n} = \tau_1 \otimes \ldots \otimes \tau_m$, the CQ φ is the following formula

$$x_{\mathbf{s}_1}, \ldots, x_{\mathbf{s}_n} \leftarrow \bigwedge_{p(t_1,\ldots,t_k) \in P} p(\mu(t_1), \ldots, \mu(t_k)),$$

where, for each constant t of the form $d_{\mathbf{s}}$ occurring in P, $\mu(t) = x_{\mathbf{s}}$. In case $K = (D, \emptyset)$ and $\varsigma(\tau) = D$ for every τ, then there are cases in which φ would explain or characterize $\mathbf{U}$, but in general this is not guaranteed. The following examples precisely show where the classical direct product breaks and provide some useful insights on how the direct product could be enriched to correctly deal with nearly connected formulas and selective knowledge bases.

Example 19. *Let us start by considering the dataset*

$$D = \{r(1,3), r(2,4), r(5,6), s(3,5), p(4,5)\},$$

which is graphically represented in Figure 4.1 as a directed graph. Let $\mathbf{U} = \{\langle 1 \rangle, \langle 2 \rangle\}$. *Accordingly,* $D \otimes D$ *is depicted in Figure 4.2. Since* $\langle 1 \rangle \otimes \langle 2 \rangle = d_{1,2}$, *the expected CQ* φ *is*

$$\begin{aligned} x_{1,2} \ \leftarrow \ & r(x_{1,1}, x_{3,3}),\ r(x_{1,2}, x_{3,4}),\ r(x_{2,2}, x_{4,4}),\ r(x_{5,1}, x_{6,3}),\ r(x_{5,5}, x_{6,6}), \\ & r(x_{2,1}, x_{4,3}),\ r(x_{2,5}, x_{4,6}),\ r(x_{1,5}, x_{3,6}),\ s(x_{3,3}, x_{5,5}),\ p(x_{4,4}, x_{5,5}), \\ & r(x_{5,2}, x_{6,4}), \end{aligned}$$

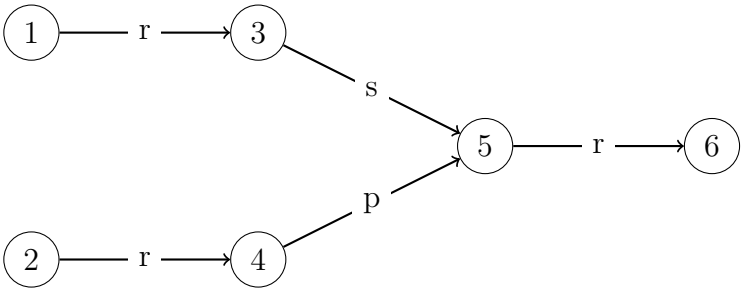

Figure 4.1: Graphical representation of dataset D presented in the Example 19.

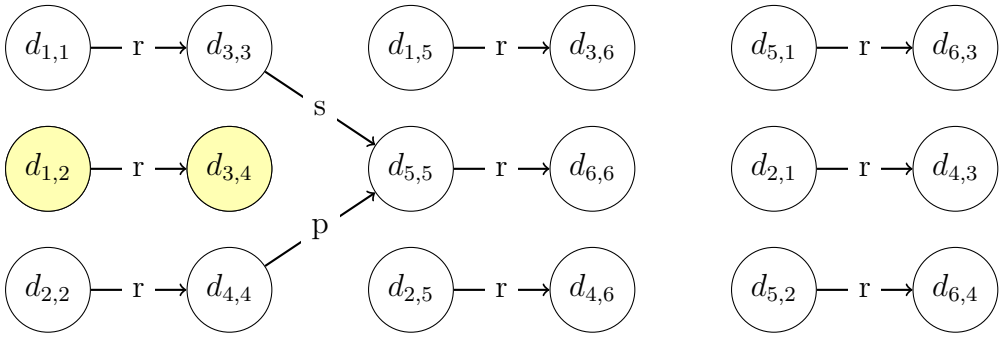

Figure 4.2: Graphical representation of dataset $D \otimes D$ presented in the Example 19.

whose atoms are isomorphic to the dataset depicted in Figure 4.2. Assume now that the given SKB $\mathcal{S} = (K, \varsigma)$ is such that $K = (D, \emptyset)$ and $\varsigma(\tau) = D^\top$ for every τ. Clearly, φ is neither a characterization nor even an explanation for $\mathbf{U}$ with respect to $\mathcal{S}$. In this case, to obtain the following characterization for $\mathbf{U}$ with respect to $\mathcal{S}$

$$x_{1,2} \leftarrow r(x_{1,2}, x_{3,4}),\ \top(x_{1,2}),\ \top(x_{3,4})$$

one can discard from φ all the atoms that are not connected to $x_{1,2}$ and add a few $\top$-atoms. Note that, the only residual atom is $r(x_{1,2}, x_{3,4})$, which is isomorphic to $r(d_{1,2}, d_{3,4})$ —the one in yellow in Figure 4.2— which, in turn, is the only one connected to $d_{1,2}$, despite the fact that D is connected.■

Unfortunately, discarding atoms not connected to free variables and adding extra $\top$-atoms is not enough to always yield to a characterization. This is illustrated via the following example.

Example 20. *Consider the SKB $\mathcal{S} = (K, \varsigma)$, where $K = (D, \emptyset)$, $D = \{r(1,3), r(2,3), r(3,4)\}$, and $\varsigma(\tau) = D^\top$ for every tuple τ. Let $\mathbf{U} = \{\langle 1\rangle, \langle 2\rangle\}$. Accordingly, we have:*

$$
\begin{array}{rcl}
D \otimes D & = & \{r(d_{1,1}, d_{3,3}),\ r(d_{2,2}, d_{3,3}),\ r(d_{3,3}, d_{4,4}),\ r(d_{1,2}, d_{3,3}),\\
& & \ r(d_{1,3}, d_{3,4}),\ r(d_{3,1}, d_{4,3}),\ r(d_{2,3}, d_{3,4}),\ r(d_{3,2}, d_{4,3}),\\
& & \ r(d_{2,1}, d_{3,3})\},\\
\varphi = x_{1,2} & \leftarrow & \ r(x_{1,1}, x_{3,3}),\ r(x_{2,2}, x_{3,3}),\ r(x_{3,3}, x_{4,4}),\ r(x_{1,2}, x_{3,3}),\\
& & \ r(x_{1,3}, x_{3,4}),\ r(x_{3,1}, x_{4,3}),\ r(x_{2,3}, x_{3,4}),\ r(x_{3,2}, x_{4,3})\\
& & \ r(x_{2,1}, x_{3,3}).
\end{array}
$$

By discarding from φ all the atoms that are not connected to $x_{1,2}$ and adding the needed $\top$-atoms, we obtain

$$
\begin{array}{rcl}
\varphi' = x_{1,2} & \leftarrow & \ r(x_{1,2}, x_{3,3}),\ r(x_{2,1}, x_{3,3}),\ r(x_{1,1}, x_{3,3}),\ r(x_{2,2}, x_{3,3}),\ r(x_{3,3}, x_{4,4}),\\
& & \ \top(x_{1,2}), \top(x_{3,3}), \top(x_{2,1}), \top(x_{1,1}), \top(x_{2,2}), \top(x_{4,4}).
\end{array}
$$

Since φ' is in NCF *and since $inst(\varphi', \mathcal{S}) \supseteq \mathbf{U}$, then φ' is an explanation for $\mathbf{U}$. However, φ' is not a characterization. Indeed, it is sufficient to notice that formula*

$$\varphi'' = x \leftarrow r(x, 3), \top(x), \top(3)$$

is also an explanation for $\mathbf{U}$ but $\varphi'' \longrightarrow \varphi'$ does not hold, contradicting Corollary 3.

A possible turn around could be the following: replace in φ' any variable of the form $x_{c,\ldots,c}$ with the constant c. This would produce the following formula

$$
\begin{array}{rcl}
\varphi''' = x_{1,2} & \leftarrow & \ r(x_{1,2}, 3),\ r(x_{2,1}, 3),\ r(1, 3),\ r(2, 3),\ r(3, 4), \top(x_{1,2}), \top(3),\\
& & \ \top(x_{2,1}), \top(1), \top(2), \top(4),
\end{array}
$$

which, this time, is a characterization for $\mathbf{U}$ (for example, $\varphi' \longrightarrow \varphi'''$ and $\varphi'' \longrightarrow \varphi'''$). ∎

Note, however, that in some cases a constant of the form $d_{c,\ldots,c}$ may give rise to a free variable $x_{c,\ldots,c}$. Clearly, if so, it cannot be replaced by c, otherwise the arity of the resulting formula would be smaller than the arity of the unit. Unfortunately, as shown by the following example, keeping the the variable $x_{c,\ldots,c}$ is not enough.

Example 21. *Consider the SKB $\mathcal{S} = (K, \varsigma)$ together with the unit $\mathbf{U} = \{\langle 1, 1\rangle, \langle 1, 2\rangle\}$, where $K = (D, \emptyset)$, $D = \{r(1, 2),\ r(2, 1),\ s(2, 1),\ s(1, 2)\}$, and ς is such that $\varsigma(\langle 1, 1\rangle) = \{r(1, 2),\ s(1, 2),\ \top(1),\ \top(2)\}$ and $\varsigma(\langle 1, 2\rangle) = \{r(1, 2),\ s(2, 1),\ \top(1),\ \top(2)\}$. In this case, since ς does not return $D^{\top}$, instead of computing $D \otimes D$ we compute $\varsigma(\langle 1, 1\rangle) \otimes \varsigma(\langle 1, 2\rangle)$ because, by Definition 7, for any characterization φ there must exist a* C*-homomorphism from $atm(\varphi)$ to both $\varsigma(\langle 1, 1\rangle)$ and $\varsigma(\langle 1, 2\rangle)$. Hence, by computing the direct product between the summaries of the tuples of $\mathbf{U}$, we obtain*

$$\begin{array}{rcl} P & = & \varsigma(\langle 1,1\rangle) \otimes \varsigma(\langle 1,2\rangle) = \\ & = & \{r(1,2), s(1,2), \top(1), \top(2)\} \otimes \{r(1,2), s(2,1), \top(1), \top(2)\} = \\ & = & \{r(d_{1,1}, d_{2,2}), s(d_{1,2}, d_{2,1}), \top(d_{1,1}), \top(d_{1,2}), \top(d_{2,1}), \top(d_{2,2})\}. \end{array}$$

Since the direct product between the tuples of $\mathbf{U}$ *is* $\langle 1,1\rangle \otimes \langle 1,2\rangle = d_{1,1}, d_{1,2}$*, then the candidate characterization (constructed by using the approach discussed before) should be:*

$$\varphi = x_{1,1}, x_{1,2} \quad \leftarrow \quad r(x_{1,1}, 2), s(x_{1,2}, x_{2,1}), \top(x_{1,1}), \top(x_{1,2}), \top(x_{2,1}), \top(2),$$

where only the constant $d_{2,2}$ *is replaced by the constant* 2*. However, this is not a characterization. Indeed, it is sufficient to notice that the formula*

$$\varphi' = x_{1,1}, x_{1,2} \quad \leftarrow \quad \top(x_{1,1}), \top(x_{1,2}), r(1,2), \top(1), \top(2)$$

is also an explanation for $\mathbf{U}$ *but* $\varphi' \longrightarrow \varphi$ *does not hold, contradicting Corollary 3.* ∎

In light of the previous example, it seems that atoms containing constants of the form $d_{c,\ldots,c}$ occurring in the sequence $\tau_1 \otimes \ldots \otimes \tau_m$ should be "cloned" so that in some of them, $d_{c,\ldots,c}$ can be replaced by $x_{c,\ldots,c}$, while in some other, $d_{c,\ldots,c}$ can be replaced by c. We are now ready to show how to construct the canonical characterization $can(\mathbf{U}, \mathcal{S})$.

Step 1 Let $d_{\mathbf{s}_1}, \ldots, d_{\mathbf{s}_n} = \tau_1 \otimes \ldots \otimes \tau_m$ denote the sequence of constants used to determine the free variables of $can(\mathbf{U}, \mathcal{S})$, and let Fr denote the set collecting these constants (note that, in general, $|Fr| \leq n$). According to Example 21, $\tau_1 = \langle 1,1\rangle$, $\tau_2 = \langle 1,2\rangle$, $d_{\mathbf{s}_1}, d_{\mathbf{s}_2} = d_{1,1}, d_{1,2}$, and Fr is the set $\{d_{1,1}, d_{1,2}\}$.

Step 2 Build the dataset $P = \varsigma(\tau_1) \otimes \ldots \otimes \varsigma(\tau_m)$ used to determine some atoms of $can(\mathbf{U}, \mathcal{S})$ (since summaries are closed under top, we also know that $Fr \subseteq \mathbb{D}_P$). See, for instance, the set P of atoms constructed in Example 21. In particular, the domain of P is $\mathbb{D}_P = \{d_{1,1}, d_{1,2}, d_{2,2}, d_{2,1}\}$.

Step 3 Consider any $d_{\mathbf{s}}$ occurring in P, and let $\mathbb{D}_{\mathbf{s}}$ be the set of constants of $\mathbf{s}$. If $|\mathbb{D}_{\mathbf{s}}| = 1$, then the atoms of P containing $d_{\mathbf{s}}$ might have to be "cloned" to determine some extra atoms of $can(\mathbf{U}, \mathcal{S})$. As we already discussed, this is needed whenever $d_{\mathbf{s}}$ satisfies both $|\mathbb{D}_{\mathbf{s}}| = 1$ and $d_{\mathbf{s}} \in Fr$. Accordingly, let $Ge = \{d_{\mathbf{s}} \in \mathbb{D}_P : |\mathbb{D}_{\mathbf{s}}| = 1\}$ be the set of constants used as possible "genes" for such clones. According to Example 21, $Ge = \{d_{1,1}, d_{2,2}\}$.

Step 4 For each $d_{\mathbf{s}} \in \mathbb{D}_P$, let

$$f(d_{\mathbf{s}}) \;=\; \begin{cases} \{d_{\mathbf{s}}, \mathbf{s}[1]\} & \text{if } d_{\mathbf{s}} \in Fr \cap Ge \\ \{d_{\mathbf{s}}\} & \text{otherwise.} \end{cases}$$

Now, for any atom $\alpha = p(t_1, \ldots, t_k)$ of P, we define

$$clones(\alpha) = \{p(c_1, \ldots, c_k) : each\ c_i \in f(t_i)\} \setminus \{\alpha\}$$

and, finally, let

$$C = \{\alpha' \in clones(\alpha) : \alpha \in P\}$$

be the set of all the clones that complement the atoms of P. According to Example 21, $Fr \cap Ge = \{d_{1,1}\}$. Moreover, $f(d_{1,1}) = \{d_{1,1}, 1\}$, $f(d_{1,2}) = \{d_{1,2}\}$, $f(d_{2,1}) = \{d_{2,1}\}$, and $f(d_{2,2}) = \{d_{2,2}\}$. Hence, $clones(\alpha) = \emptyset$, whenever $\alpha \in \{s(d_{1,2}, d_{2,1}),\ \top(d_{1,2}),\ \top(d_{2,1}),\ \top(d_{2,2})\}$; $clones(r(d_{1,1}, d_{2,2})) = \{r(1, d_{2,2})\}$; and $clones(\top(d_{1,1})) = \{\top(1)\}$. Finally, we have that $C = \{r(1, d_{2,2}),\ \top(1)\}$ and the domain of C is $\mathbb{D}_C = \{1,\ d_{2,2}\}$.

Step 5 Let μ be the mapping $\{c \mapsto c : c \in \mathbb{D}_C \setminus \mathbb{D}_P\} \;\cup\; \{d_{\mathbf{s}} \mapsto g(d_{\mathbf{s}}) : d_{\mathbf{s}} \in \mathbb{D}_P\}$ used to transform atoms of $P \cup C$ into atoms of $can(\mathbf{U}, \mathcal{S})$, where

$$g(d_{\mathbf{s}}) \;=\; \begin{cases} x_{\mathbf{s}} & \text{if } d_{\mathbf{s}} \in Fr \\ y_{\mathbf{s}} & \text{if } d_{\mathbf{s}} \notin Fr \;\cup\; Ge \\ \mathbf{s}[1] & \text{if } d_{\mathbf{s}} \in Ge \setminus Fr. \end{cases}$$

Consider the next formula ("$\wedge$" is used instead of ","):

$$\Phi(\mathbf{U}, \mathcal{S}) = x_{\mathbf{s}_1}, \ldots, x_{\mathbf{s}_n} \leftarrow \bigwedge_{p(t_1,\ldots,t_k) \in P \,\cup\, C} p(\mu(t_1), \ldots, \mu(t_k)).$$

According to Example 21, $Fr \cup Ge = \{d_{1,1}, d_{1,2}, d_{2,2}\}$ and $Ge \setminus Fr = \{d_{2,2}\}$. Hence, $g(d_{1,1}) = x_{1,1}$, $g(d_{1,2}) = x_{1,2}$, $g(d_{2,1}) = y_{2,1}$, and $g(d_{2,2}) = 2$. Moreover, $\mathbb{D}_C \setminus \mathbb{D}_P = \{1\}$ and, therefore, $\mu = \{1 \mapsto 1\} \;\cup\; \{d_{1,1} \mapsto x_{1,1},\ d_{1,2} \mapsto x_{1,2},\ d_{2,1} \mapsto y_{2,1},\ d_{2,2} \mapsto 2\}$. Then, we have

$$P \;\cup\; C = \{r(d_{1,1},\ d_{2,2}),\ s(d_{1,2},\ d_{2,1}),\ \top(d_{1,1}),\ \top(d_{1,2}),\ \top(d_{2,1}),\ \top(d_{2,2}),\ r(1, d_{2,2}),\ \top(1)\}.$$

Finally, we obtain

$$\Phi(\mathbf{U}, \mathcal{S}) = x_{1,1}, x_{1,2} \;\leftarrow\; r(x_{1,1}, 2), s(x_{1,2}, y_{2,1}), \top(x_{1,1}), \top(x_{1,2}), \top(y_{2,1}), \top(2), r(1, 2), \top(1).$$

Step 6 We are now ready to define the desired canonical characterization $can(\mathbf{U}, \mathcal{S})$.

Definition 9. *We define* $can(\mathbf{U}, \mathcal{S})$ *as the nearly connected formula obtained from* $\Phi(\mathbf{U}, \mathcal{S})$ *by discarding all and only the atoms that are not connected to any free variable of* $\Phi(\mathbf{U}, \mathcal{S})$*. We refer to* $can(\mathbf{U}, \mathcal{S})$ *as the* canonical characterization *of* $\mathbf{U}$ *according to* $\mathcal{S}$*.* ∎

According to Example 21, since $\Phi(\mathbf{U}, \mathcal{S})$ is already a nearly connected formula, then we immediately get that $can(\mathbf{U}, \mathcal{S}) = \Phi(\mathbf{U}, \mathcal{S})$. What remains to do is showing that $can(\mathbf{U}, \mathcal{S})$ is always a characterization for $\mathbf{U}$ with respect to $\mathcal{S}$.

Proposition 9. *It holds that* $can(\mathbf{U}, \mathcal{S})$ *characterizes* $\mathbf{U}$*.*

Proof. For the rest of the proof let $\varphi = can(\mathbf{U}, \mathcal{S})$ and $\Phi = \Phi(\mathbf{U}, \mathcal{S})$; the latter always exists in light of Corollary 4. In order for φ to characterize $\mathbf{U}$, by Definition 7, it must satisfy two requirements. (i) it must explain $\mathbf{U}$ and (ii) $\varphi \longrightarrow \varphi'$ must imply that $\varphi' \longrightarrow \varphi$ whenever φ' explains $\mathbf{U}$. Point (i) will be shown directly, while for point (ii), we will follow Corollary 3, hence we will show a stronger result, i.e. $\varphi' \longrightarrow \varphi$ whenever φ' explains $\mathbf{U}$ regardless of $\varphi \longrightarrow \varphi'$. We will start by showing point (i). In order to do so, since $\varphi \longrightarrow \Phi$, if we show that Φ explains $\mathbf{U}$ then by C-homomorphism composition we are done. Hence, it suffices to show that $inst(\Phi, \mathcal{S}) \supseteq \mathbf{U}$. To this aim, first notice that, by construction, μ acts as a bijection between $atm(\Phi)$ and $P \cup C$, moreover μ^{-1} does act as a homomorphism from $atm(\Phi)$ to $P \cup C$ since by costruction it preserves every atom. Define now $h_i = \{a \mapsto a \ : \ a \in \mathbb{D}_D\} \cup \{d_{\mathbf{s}} \mapsto \mathbf{s}[i] \ : \ d_{\mathbf{s}} \in \mathbb{D}_P\}$. By construction, each h_i acts as the projection on the i-th component. Hence, each h_i is a homomorphism from $P \cup C$ to $\varsigma(\tau_i)$. Let now $A = \mathbb{D}_C \setminus \mathbb{D}_P$. Since, by construction, $\mu^{-1}(c)$ is either c when $c \in A$ or of the form $d_{c,\dots,c}$ otherwise, and since $h_i(c) = c$ and $h_i(d_{c,\dots,c}) = c$, it holds that each $\lambda_i = h_i \circ \mu^{-1}$ is a C-homomorphism from $atm(\Phi)$ to $\varsigma(\tau_i)$. In particular, by construction, we have $\langle \lambda_i(x_{\mathbf{s}_1}), \dots, \lambda_i(x_{\mathbf{s}_n}) \rangle = \tau_i$. Therefore, $inst(\Phi, \mathcal{S}) \supseteq \mathbf{U}$, and this concludes the proof of point (i). Now, in order to get point (ii) we will prove $\varphi' \longrightarrow \varphi$ whenever φ' explains $\mathbf{U}$ regardless of $\varphi \longrightarrow \varphi'$. To this aim, first notice that if $\varphi' \longrightarrow \varphi''$, with $\varphi' \in \mathrm{N\mathbb{C}F}$, we get that $\varphi' \longrightarrow \tilde{\varphi}$, where $\tilde{\varphi}$ is the nearly connected formula obtained from φ'' by discarding all and only the atoms that are not connected to any free variable of φ''. This is true as homomorphisms preserve the connection property of structures in general. Since φ is the nearly connected part of Φ, we will just show that $\varphi' \longrightarrow \Phi$ whenever φ' explains $\mathbf{U}$. This, together with

the fact that homomorphisms do preserve connectedness, will give us the result we want, since if φ' explains $\mathbf{U}$ then, by Definition 6, in particular $\varphi' \in \mathrm{N}\mathbb{C}\mathrm{F}$. Consider now an arbitrary formula φ' that explains $\mathbf{U}$ and let $z_1, \ldots, z_n$ be its free variables. Since, by definition, $inst(\varphi', \mathcal{S}) \supseteq \mathbf{U}$ holds, for each $i \in \{1, \ldots, m\}$, we can consider a homomorphism h_i that maps $atm(\varphi')$ to $\varsigma(\tau_i)$ such that $\langle h_i(z_1), \ldots, h_i(z_n)\rangle = \tau_i$. Let us now also consider the following mapping $\xi = \{t \mapsto d_{h_1(t),\ldots,h_m(t)} \ : \ t \in \mathbb{D}(\varphi') \setminus A\} \cup \{t \mapsto t \ : \ t \in \mathbb{D}(\varphi') \cap A\}$. ξ acts as a homomorphism from $atm(\varphi')$ to $P \cup C$ since it preserves every atom. Considering now $h = \mu \circ \xi$, we get a C-homomorphism from $atm(\varphi')$ to $atm(\Phi)$ such that each $h(z_i)$ equals to $x_{\mathbf{s}_i}$. In order to see that h is a C-homomorphism it suffices to see how constants do behave, in particular we will first map $c \in \mathsf{C}$ to $d_{c,\ldots,c}$ and then from $d_{c,\ldots,c}$ to c again. Now our proof is completed. □

Since $can(\mathbf{U}, \mathcal{S})$ is always constructible for any given unit and any given SKB, it is clear now that the truthfulness of Theorem 1 directly follows by Proposition 9. For completeness of exposition, we close the section by showing how to systematically construct $can(\mathbf{U}_0, \mathcal{S}_0)$ according to our running example started in the Introduction.

Example 22. *According to our Example 1, the direct product of the elements of* $\mathbf{U}_0$ *is the (unary) sequence* $\langle\mathsf{Epcot}\rangle \otimes \langle\mathsf{Discovery\_Cove}\rangle$ *that is equal to* $d_{\mathsf{Epcot},\mathsf{Discovery\_Cove}}$. *Hereinafter, to lighten the notation, we denote* $\mathsf{Discovery\_Cove}$ *by* D, Epcot *by* E, *and* $\mathsf{Florida}$ *by* F. *Thus,* $Fr = \{d_{\mathsf{E},\mathsf{D}}\}$. *Now, to compute* $P = \varsigma_0(\langle\mathsf{Epcot}\rangle) \otimes \varsigma_0(\langle\mathsf{Discovery\ Cove}\rangle)$, *we need exploit the summaries of* $\langle\mathsf{Epcot}\rangle$ *and* $\langle\mathsf{Discovery\_Cove}\rangle$ *already introduced In Example 15:*

$$\begin{aligned}
\varsigma_0(\langle\mathsf{E}\rangle) &= \{\mathsf{located}(\mathsf{E},\mathsf{F}), \mathsf{partOf}(\mathsf{F},\mathsf{US}), \mathsf{isa}(\mathsf{E},\mathsf{tp}), \mathsf{isa}(\mathsf{E},\mathsf{ap}),\\
&\quad \top(\mathsf{E}), \top(\mathsf{F}), \top(\mathsf{US}), \top(\mathsf{tp}), \top(\mathsf{ap})\}\\
\varsigma_0(\langle\mathsf{D}\rangle) &= \{\mathsf{located}(\mathsf{D},\mathsf{F}), \mathsf{partOf}(\mathsf{F},\mathsf{US}), \mathsf{isa}(\mathsf{D},\mathsf{tp}), \mathsf{isa}(\mathsf{D},\mathsf{ap}),\\
&\quad \top(\mathsf{D}), \top(\mathsf{F}), \top(\mathsf{US}), \top(\mathsf{tp}), \top(\mathsf{ap})\}
\end{aligned}$$

Therefore, the set P *of atoms is:*

$$\begin{aligned}
P = \{&\mathsf{located}(d_{\mathsf{E},\mathsf{D}}, d_{\mathsf{F},\mathsf{F}}), \mathsf{partOf}(d_{\mathsf{F},\mathsf{F}}, d_{\mathsf{US},\mathsf{US}}), \mathsf{isa}(d_{\mathsf{E},\mathsf{D}}, d_{\mathsf{tp},\mathsf{tp}}),\\
&\mathsf{isa}(d_{\mathsf{E},\mathsf{D}}, d_{\mathsf{tp},\mathsf{ap}}), \mathsf{isa}(d_{\mathsf{E},\mathsf{D}}, d_{\mathsf{ap},\mathsf{tp}}), \mathsf{isa}(d_{\mathsf{E},\mathsf{D}}, d_{\mathsf{ap},\mathsf{ap}}), \top(d_{\mathsf{E},\mathsf{D}}),\\
&\top(d_{\mathsf{E},\mathsf{F}}), \top(d_{\mathsf{E},\mathsf{US}}), \top(d_{\mathsf{E},\mathsf{tp}}), \top(d_{\mathsf{E},\mathsf{ap}}), \top(d_{\mathsf{F},\mathsf{D}}), \top(d_{\mathsf{F},\mathsf{F}}),\\
&\top(d_{\mathsf{F},\mathsf{US}}), \top(d_{\mathsf{F},\mathsf{tp}}), \top(d_{\mathsf{F},\mathsf{ap}}), \top(d_{\mathsf{tp},\mathsf{D}}), \top(d_{\mathsf{tp},\mathsf{F}}), \top(d_{\mathsf{tp},\mathsf{US}}),\\
&\top(d_{\mathsf{tp},\mathsf{tp}}), \top(d_{\mathsf{tp},\mathsf{ap}}), \top(d_{\mathsf{ap},\mathsf{ap}}), \top(d_{\mathsf{ap},\mathsf{D}}), \top(d_{\mathsf{ap},\mathsf{F}}), \top(d_{\mathsf{ap},\mathsf{US}}),\\
&\top(d_{\mathsf{ap},\mathsf{tp}}), \top(d_{\mathsf{US},\mathsf{tp}}), \top(d_{\mathsf{US},\mathsf{D}}), \top(d_{\mathsf{US},\mathsf{F}}), \top(d_{\mathsf{US},\mathsf{US}}), \top(d_{\mathsf{US},\mathsf{ap}})\}.
\end{aligned}$$

Accordingly, $Ge = \{d_{\mathsf{F},\mathsf{F}}, d_{\mathsf{US},\mathsf{US}}, d_{\mathsf{tp},\mathsf{tp}}, d_{\mathsf{ap},\mathsf{ap}}\}$. *This time, since* $Fr \cap Ge = \emptyset$, *we have that also the set* C *is empty. Therefore,*

$$\begin{aligned}\mu \;=\; \{&d_{\mathsf{E,D}} \mapsto x_{\mathsf{E,D}}, d_{\mathsf{E,F}} \mapsto y_{\mathsf{E,F}}, d_{\mathsf{E,US}} \mapsto y_{\mathsf{E,US}}, d_{\mathsf{E,tp}} \mapsto y_{\mathsf{E,tp}},\\ &d_{\mathsf{E,ap}} \mapsto y_{\mathsf{E,ap}}, d_{\mathsf{F,D}} \mapsto y_{\mathsf{F,D}}, d_{\mathsf{F,F}} \mapsto \mathsf{F}, d_{\mathsf{F,US}} \mapsto y_{\mathsf{F,US}},\\ &d_{\mathsf{F,tp}} \mapsto y_{\mathsf{F,tp}}, d_{\mathsf{F,ap}} \mapsto y_{\mathsf{F,ap}}, d_{\mathsf{tp,D}} \mapsto y_{\mathsf{tp,D}}, d_{\mathsf{tp,F}} \mapsto y_{\mathsf{tp,F}},\\ &d_{\mathsf{tp,US}} \mapsto y_{\mathsf{tp,US}}, d_{\mathsf{tp,tp}} \mapsto \mathsf{tp}, d_{\mathsf{tp,ap}} \mapsto y_{\mathsf{tp,ap}}, d_{\mathsf{ap,ap}} \mapsto \mathsf{ap},\\ &d_{\mathsf{ap,D}} \mapsto y_{\mathsf{ap,D}}, d_{\mathsf{ap,F}} \mapsto y_{\mathsf{ap,F}}, d_{\mathsf{ap,US}} \mapsto y_{\mathsf{ap,US}}, d_{\mathsf{ap,tp}} \mapsto y_{\mathsf{ap,tp}},\\ &d_{\mathsf{US,tp}} \mapsto y_{\mathsf{US,tp}}, d_{\mathsf{US,D}} \mapsto y_{\mathsf{US,D}}, d_{\mathsf{US,F}} \mapsto y_{\mathsf{US,F}}, d_{\mathsf{US,US}} \mapsto \mathsf{US},\\ &d_{\mathsf{US,ap}} \mapsto y_{\mathsf{US,ap}}\}.\end{aligned}$$

Thus,

$$\begin{aligned}\Phi(\mathbf{U}_0, \mathcal{S}_0) = x_{\mathsf{E,D}} \;\leftarrow\; &\mathsf{located}(x_{\mathsf{E,D}}, \mathsf{F}), \mathsf{partOf}(\mathsf{F}, \mathsf{US}), \mathsf{isa}(x_{\mathsf{E,D}}, \mathsf{tp}),\\ &\mathsf{isa}(x_{\mathsf{E,D}}, y_{\mathsf{tp,ap}}), \mathsf{isa}(x_{\mathsf{E,D}}, y_{\mathsf{ap,tp}}), \mathsf{isa}(x_{\mathsf{E,D}}, \mathsf{ap}),\\ &\top(x_{\mathsf{E,D}}), \top(y_{\mathsf{E,F}}), \top(y_{\mathsf{E,US}}), \top(y_{\mathsf{E,tp}}), \top(y_{\mathsf{E,ap}}),\\ &\top(y_{\mathsf{F,D}}), \top(\mathsf{F}), \top(y_{\mathsf{F,US}}), \top(y_{\mathsf{F,tp}}), \top(y_{\mathsf{F,ap}}),\\ &\top(y_{\mathsf{tp,D}}), \top(y_{\mathsf{tp,F}}), \top(y_{\mathsf{tp,US}}), \top(\mathsf{tp}), \top(y_{\mathsf{tp,ap}}), \top(\mathsf{ap}),\\ &\top(y_{\mathsf{ap,D}}), \top(y_{\mathsf{ap,F}}), \top(y_{\mathsf{ap,US}}), \top(y_{\mathsf{ap,tp}}), \top(y_{\mathsf{US,tp}}),\\ &\top(y_{\mathsf{US,D}}), \top(y_{\mathsf{US,F}}), \top(\mathsf{US}), \top(y_{\mathsf{US,ap}}).\end{aligned}$$

Now, we can obtain the canonical characterization

$$\begin{aligned}can(\mathbf{U}_0, \mathcal{S}_0) = x_{\mathsf{E,D}} \;\leftarrow\; &\mathsf{located}(x_{\mathsf{E,D}}, \mathsf{F}), \mathsf{partOf}(\mathsf{F}, \mathsf{US}), \mathsf{isa}(x_{\mathsf{E,D}}, \mathsf{tp}),\\ &\mathsf{isa}(x_{\mathsf{E,D}}, y_{\mathsf{tp,ap}}), \mathsf{isa}(x_{\mathsf{E,D}}, y_{\mathsf{ap,tp}}), \mathsf{isa}(x_{\mathsf{E,D}}, \mathsf{ap}),\\ &\top(x_{\mathsf{E,D}}), \top(\mathsf{F}), \top(y_{\mathsf{ap,tp}}), \top(\mathsf{US}), \top(\mathsf{tp}),\\ &\top(y_{\mathsf{tp,ap}}), \top(\mathsf{ap}),\end{aligned}$$

by discarding all the atoms that are not connected to the free variable $x_{\mathsf{E,D}}$.
■

4.4 Core characterizations

One can observe that $can(\mathbf{U}_0, \mathcal{S}_0)$ constructed in Example 22 and $\bar{\varphi}_a$ given in Figure 1.2 are (homomorphically) equivalent. Indeed, $\bar{\varphi}_a \longrightarrow can(\mathbf{U}_0, \mathcal{S}_0)$ holds via the C-homomorphism $h = \{x \mapsto x_{\mathsf{E,D}}, \mathsf{F} \mapsto \mathsf{F}, \mathsf{US} \mapsto \mathsf{US}, \mathsf{tp} \mapsto \mathsf{tp}, \mathsf{ap} \mapsto \mathsf{ap}\}$; moreover, $can(\mathbf{U}_0, \mathcal{S}_0) \longrightarrow \bar{\varphi}_a$ holds via the C-homomorphism $h' = \{x_{\mathsf{E,D}} \mapsto x, \mathsf{F} \mapsto \mathsf{F}, y_{\mathsf{ap,tp}} \mapsto \mathsf{ap}, \mathsf{US} \mapsto \mathsf{US}, \mathsf{tp} \mapsto \mathsf{tp}, y_{\mathsf{tp,ap}} \mapsto \mathsf{ap}, \mathsf{ap} \mapsto \mathsf{ap}\}$. Hence, $\bar{\varphi}_a \longleftrightarrow can(\mathbf{U}_0, \mathcal{S}_0)$ holds and, as a result, we can conclude that $\bar{\varphi}_a$ also characterizes $\mathbf{U}_0$. However, it is easy to verify that $\bar{\varphi}_a$ is smaller (with respect to the number of atoms) that $can(\mathbf{U}_0, \mathcal{S}_0)$ (9 vs. 13 atoms).

One can wonder whether there is a systematic way to construct characterizations that are as small as possible. To this aim, consider two formulas φ_1 and φ_2. If $\varphi_1 \longleftrightarrow \varphi_2$ holds in virtue of a bijective C-homomorphism, then the formulas are said *isomorphic*, denoted by $\varphi_1 \simeq \varphi_2$, where $\simeq$ is

an *equivalence relation*. Accordingly, given a formula φ, $[\varphi]$ denotes the *equivalence class* of φ collecting all formulas φ' such that $\varphi \simeq \varphi'$.

Coming back to Example 22, it can be easily checked that $can(\mathbf{U}_0, \mathcal{S}_0)$ is isomorphic to

$$x \leftarrow conj(\bar{\varphi}_a), \mathsf{isa}(x, y_1), \mathsf{isa}(x, y_2), \top(y_1), \top(y_2),$$

where $\bar{\varphi}_a$ is the formula reported in Figure 1.2 and $h'' = \{x_{\mathsf{E,D}} \mapsto x, \mathsf{F} \mapsto \mathsf{F}, y_{\mathsf{ap,tp}} \mapsto y_1, \mathsf{US} \mapsto \mathsf{US}, \mathsf{tp} \mapsto \mathsf{tp}, y_{\mathsf{tp,ap}} \mapsto y_2, \mathsf{ap} \mapsto \mathsf{ap}\}$ is the isomorphism between the two formulas. Therefore, the atoms after $conj(\bar{\varphi}_a)$ can be considered redundant, since just $\bar{\varphi}_a$ is a characterization.

To get rid of redundant atoms, one could exploit the notion of core of a structure (see Section 2.2) and lift it to formulas, in the same spirit as done in [78] for conjunctive queries (namely, CFs with no constant).

Definition 10. *Consider a formula φ. A core of φ is any formula φ' with $atm(\varphi) \subseteq atm(\varphi')$ such that $\varphi' \longrightarrow \varphi$ and there is no φ'' with $atm(\varphi'') \subset atm(\varphi')$ such that $\varphi' \longrightarrow \varphi''$. Accordingly, any formula isomorphic to φ' is also a core of φ and it will be denoted by $core(\varphi)$.* ∎

According to the previous definition, $core(\varphi) \longleftrightarrow \varphi$. Moreover, $\varphi_1 \longleftrightarrow \varphi_2$ if, and only if, $[core(\varphi_1)] = [core(\varphi_2)]$.

Example 23. *Consider the following two formulas*

$$\begin{array}{rcl} \bar{\varphi}_6 = x & \leftarrow & \mathsf{isa}(x, \mathsf{tp}), \mathsf{located}(x, y), \mathsf{located}(x, \mathsf{Florida}) \\ \bar{\varphi}_7 = z & \leftarrow & \mathsf{isa}(z, \mathsf{tp}), \mathsf{located}(z, \mathsf{Florida}) \end{array}$$

We get that $\bar{\varphi}_6 \longleftrightarrow \bar{\varphi}_7$. Indeed, we can consider the C*-homomorphism $h_1 = \{x \mapsto z, \mathsf{tp} \mapsto \mathsf{tp}, y \mapsto \mathsf{Florida}, \mathsf{Florida} \mapsto \mathsf{Florida}\}$ from $atm(\bar{\varphi}_6)$ to $atm(\bar{\varphi}_7)$ and the* C*-homomorphism $h_2 = \{z \mapsto x, \mathsf{tp} \mapsto \mathsf{tp}, \mathsf{Florida} \mapsto \mathsf{Florida}\}$ from $atm(\bar{\varphi}_7)$ to $atm(\bar{\varphi}_6)$. Hence, $[core(\bar{\varphi}_7)] = [core(\bar{\varphi}_6)]$ and, in particular, $[core(\bar{\varphi}_7)] = [\bar{\varphi}_7]$. The latter holds since there cannot exist a formula equivalent to $\bar{\varphi}_7$ that is smaller in size due to the existence of* tp *and* Florida *in the two distinct atoms of $atm(\bar{\varphi}_7)$.* ∎

Note that, as well-known from database theory, the core of a structure (and, therefore, of a conjunctive formula) can be dramatically smaller than the structure (resp., formula) itself. As an example, it suffices to think about the infinite family of datasets encoding —via some binary relation *arc*— all 3-colorable graphs that also contain arcs of the form $\mathsf{Tr} = \{arc(c_1, c_2), arc(c_2, c_1), arc(c_2, c_3), arc(c_3, c_2), arc(c_3, c_1), arc(c_1, c_3)\}$, it is evident that the set Tr represents a "triangle". Indeed, the cores of all these graphs coincide to Tr itself. The notion of core could be applied on canonical characterizations to reduce their sizes. But to do that, we first need the following results.

Proposition 10. *The core of a nearly connected formula is still nearly connected.*

Proof. Let φ be a nearly connected formula, and consider $core(\varphi)$. First of all, by Definition 10, we know that $\varphi \longrightarrow core(\varphi)$. Thus, let h be a C-homomorphism from $atm(\varphi)$ to $atm(core(\varphi))$, and assume, without loss of generality, that the free variables of φ coincide with the free variables of $core(\varphi)$, namely $x_1, \ldots, x_n$. Hence, $h(x_i) = x_i$, for each $i = 1, \ldots, n$. Therefore, h is also a C-homomorphism from $atm(\varphi) \cup \{\mathsf{free}(x_1, \ldots, x_n)\}$ to $atm(core(\varphi)) \cup \{\mathsf{free}(x_1, \ldots, x_n)\}$. Second, by Definition 10, $atm(core(\varphi)) \subseteq atm(\varphi)$. Hence, adding to both members $\{\mathsf{free}(x_1, \ldots, x_n)\}$, we get that $atm(core(\varphi)) \cup \{\mathsf{free}(x_1, \ldots, x_n)\} \subseteq atm(\varphi) \cup \{\mathsf{free}(x_1, \ldots, x_n)\}$. Third, by Definition 10, there is no φ' with $atm(\varphi') \subset atm(core(\varphi))$ such that $core(\varphi) \longrightarrow \varphi'$. Hence, there is no structure S such that $S \subset atm(core(\varphi)) \cup \{\mathsf{free}(x_1, \ldots, x_n)\}$ that admits a C-homomorphism from $atm(core(\varphi)) \cup \{\mathsf{free}(x_1, \ldots, x_n)\}$ to S. Therefore, we have proved that $atm(core(\varphi)) \cup \{\mathsf{free}(x_1, \ldots, x_n)\}$ is the core of $atm(\varphi) \cup \{\mathsf{free}(x_1, \ldots, x_n)\}$. Now, we know that φ is nearly connected if, and only if, $atm(\varphi) \cup \{\mathsf{free}(x_1, \ldots, x_n)\}$ is a connected structure. Moreover, it is well known that the core of a connected structure is also connected. Hence, $atm(core(\varphi)) \cup \{\mathsf{free}(x_1, \ldots, x_n)\}$ is a connected structure. Therefore, $core(\varphi)$ is a nearly connected formulas. □

According to the above Proposition, the next result immediately holds.

Corollary 5. *The core of a characterization is still a characterization.*

We are now ready to define characterizations of minimal size.

Definition 11. *We define* $core(\mathbf{U}, \mathcal{S})$ *as the core —up to isomorphism— of* $can(\mathbf{U}, \mathcal{S})$*, and we refer to it as the* core characterization *of* $\mathbf{U}$ *according to* $\mathcal{S}$*.* ■

Let us illustate the above notion with an example.

Example 24. *Concerning our running example, we can state that* $\bar{\varphi}_a \simeq core(\mathbf{U}_0, \mathcal{S}_0)$*. In particular, it can be noticed that* $\bar{\varphi}_a \simeq core(\bar{\varphi}_a)$*. To this end, we can show that there exists no* C*-homomorphism from* $atm(\bar{\varphi}_a)$ *to a proper subset of itself. To conclude this, it is sufficient to note that, for each atom* α *in* $atm(\bar{\varphi}_a)$*, we get either* $\mathbb{D}_{\{\alpha\}} \cap \mathsf{V} = \{x\}$ *or* $\mathbb{D}_{\{\alpha\}} \cap \mathsf{V} = \emptyset$*. In the latter case, it is evident that the atom may not be elided since any* C*-homomorphism must preserve elements from* C *and* $\mathbb{D}_{\{\alpha\}} \subseteq \mathsf{C} \cup \mathsf{V}$*. In the former case, on the other hand, the only thing that a* C*-homomorphism can do is a renaming of* x*. This, together with the fact that* $conj(\bar{\varphi}_a)$ *contains no repeated atoms, gives us* $\bar{\varphi}_a \simeq core(\bar{\varphi}_a)$*.* ■

We close the section with the most interesting properties of core characterizations.

Theorem 2. *Core characterizations always exist and are of minimal size.*

Proof. Before diving headlong into the passages below we give an outline of the proof. We will divide the argument in two part, (a) existence of core characterization and (b) minimality of their size.

In order to prove (a), first we will show that (i) given any formula it always admits a core. Given point (i) we will exploit Theorem 1, from which we will get that a characterization, call it φ, always exists, together with Corollary 5, from which we will get that $core(\varphi)$ is still a characterization, and this will end the first part of the proof.

In order to prove (b), we will use Proposition 7 to show by contradiction that is impossible that there exist a characterization smaller in size with respect to one that is a core.

We start with the (a) part of the proof. To prove point (i) of part (a) consider a generic formula φ. Now we have two possibilities, either $\varphi = core(\varphi)$, or, by Definition 10, it exists a formula φ', with $atm(\varphi') \subset atm(\varphi)$, such that $\varphi \longrightarrow \varphi'$. We can now apply the same argument to φ'. In fact, either $\varphi' = core(\varphi)$, or, by Definition 10, there exists a formula φ'', with $atm(\varphi'') \subset atm(\varphi')$, such that $\varphi' \longrightarrow \varphi''$. Since starting from a formula φ of size m the number of possible formulas φ' to check is at most 2^m, after a finite number of steps we will find a core of φ. Now part (a), as we anticipated, follows exploiting Theorem 1, from which we get that a characterization always exists, together with Corollary 5, from which we get that its core is still a characterization.

We still have to show the part (b). To this end, we consider two generic characterizations for the same unit $\mathbf{U}$, call them φ and φ' respectively, and suppose by contradiction that $\varphi = core(\varphi)$ and that $|\varphi'| < |\varphi|$. By Proposition 7 we know that $\varphi \longleftrightarrow \varphi'$, but by C-homomorphism composition, since $|\varphi'| < |\varphi|$, we know that there exist $\tilde{\varphi}$ such that $atm(\tilde{\varphi}) \subset atm(\varphi)$ and $\varphi' \longrightarrow \tilde{\varphi}$. However, since $\varphi \longrightarrow \varphi' \longrightarrow \tilde{\varphi}$, again by C-homomorphism composition the existence of $\tilde{\varphi}$ goes against Definition 10, hence it goes against the fact that $\varphi = core(\varphi)$, that is absurd. □

Chapter 5

Taxonomic Expansions

In this chapter, we aim to illuminate a novel and inherently more intuitive methodology for the generalization of sets of tuples of entities. To this end, we shall expound upon the concept of the Expansion Graph. This pivotal construct will enable us to elucidate the intrinsic properties of the expansions of sets of tuples of entities, advocating for a taxonomic approach to their treatment as opposed to the historically prevalent linear methodology. In what follows, consider to have an SKB $\mathcal{S} = (K, \varsigma)$ with $K = (D, O)$ together with an n-ary $\mathcal{S}$-unit $\mathbf{U} = \{\tau_1, \ldots, \tau_m\}$. Let $\mathbf{T}$ be the set of all n-ary tuples over $\mathbb{D}_D$. The ultimate goal of this section is to be able to classify each tuple τ of $\mathbf{T}$ in relation to $\mathbf{U}$, by characterizing each unit $\mathbf{U} \cup \{\tau\}$. This will lead to the *expansion graph* of $\mathbf{U}$ with respect to $\mathcal{S}$, denoted by $eg(\mathbf{U}, \mathcal{S})$.

5.1 Essential Expansions

Towards our objective, we first observe that characterizations enable the classification of every tuple τ in $\mathbf{T}$ in relation to $\mathbf{U}$. Specifically, the formula $core(\mathbf{U} \cup \{\tau\}, \mathcal{S})$ provides a concise explanation of all and only the nexus of similarity between τ and the tuples of $\mathbf{U}$ with respect to $\mathcal{S}$ according to N$\mathbb{C}$Fs. In simpler terms, $core(\mathbf{U} \cup \{\tau\}, \mathcal{S})$ represents the most concise least general generalization of $core(\mathbf{U}, \mathcal{S})$ that explains both $\mathbf{U}$ and $\{\tau\}$. The sets of instances of these formulas are the expected expansions of $\mathbf{U}$. This is formalized next.

Definition 12. *The* essential expansion *of the $\mathcal{S}$-unit $\mathbf{U}$, denoted by $ess(\mathbf{U}, \mathcal{S})$, is the set $inst(\varphi, \mathcal{S})$, where $\varphi = core(\mathbf{U}, \mathcal{S})$.*[1] ∎

[1] Here φ might be equivalently replaced by any other $\mathcal{S}$-characterization of $\mathbf{U}$.

Example 25. *Consider again our running example. We have seen that* $\bar{\varphi}_a \simeq core(\mathbf{U}_0, \mathcal{S}_0)$*. Hence, the following set*

$$ess(\mathbf{U}_0, \mathcal{S}_0) = inst(\bar{\varphi}_a, \mathcal{S}_0) = \{\langle \mathsf{Discovery\_Cove} \rangle, \langle \mathsf{Epcot} \rangle\}$$

is the essential expansion of the $\mathcal{S}_0$*-unit* $\mathbf{U}_0$*.* ∎

Clearly, $ess(\mathbf{U}, \mathcal{S}) \supseteq \mathbf{U}$ always holds. Interestingly,

Proposition 11. *Consider some* $\mathcal{S}$*-unit* $\mathbf{U}' \supseteq \mathbf{U}$*. It holds that* $core(\mathbf{U}, \mathcal{S})$ *characterizes* $\mathbf{U}'$ *if, and only if,* $\mathbf{U}' \subseteq ess(\mathbf{U}, \mathcal{S})$*.*

Proof. First, assume that $\mathbf{U}' \subseteq ess(\mathbf{U}, \mathcal{S})$. By Definition 12, $ess(\mathbf{U}, \mathcal{S}) = inst(core(\mathbf{U}, \mathcal{S}), \mathcal{S})$. Thus, by Definition 6, $core(\mathbf{U}, \mathcal{S})$ explains $\mathbf{U}'$. Now, by Corollary 3, we have that $core(\mathbf{U}, \mathcal{S}) \longrightarrow core(\mathbf{U}', \mathcal{S})$. At the same time, since $core(\mathbf{U}', \mathcal{S})$ explains $\mathbf{U}$, again by Corollary 3, $core(\mathbf{U}', \mathcal{S}) \longrightarrow core(\mathbf{U}, \mathcal{S})$. In particular, since both of them are cores, we get $core(\mathbf{U}', \mathcal{S}) \simeq core(\mathbf{U}, \mathcal{S})$. Therefore, $core(\mathbf{U}, \mathcal{S})$ characterizes $\mathbf{U}'$.

Now, assume that $\mathbf{U}' \not\subseteq ess(\mathbf{U}, \mathcal{S})$. Then, since $inst(core(\mathbf{U}, \mathcal{S}), \mathcal{S}) = ess(\mathbf{U}, \mathcal{S})$, it holds that $core(\mathbf{U}, \mathcal{S})$ does not explain $\mathbf{U}'$. Hence, by following Definition 7, $core(\mathbf{U}, \mathcal{S})$ does not characterize $\mathbf{U}'$. □

The proposition above essentially confirms that $ess(\mathbf{U}, \mathcal{S})$ represents the minimal expected expansion of $\mathbf{U}$ with respect to $\mathcal{S}$. In other words, all the tuples in $ess(\mathbf{U}, \mathcal{S}) \setminus \mathbf{U}$ have the same nexus of similarity with $\mathbf{U}$. Analogously, $ess(\mathbf{U} \cup \{\tau\}, \mathcal{S})$ represents the essential expansion of $\mathbf{U} \cup \{\tau\}$. By comparing all such expansions in the light of Corollary 3, we can derive the following result.

Proposition 12. *Let* τ_1 *and* τ_2 *be two tuples in* $\mathbf{T}$*. Then it holds that* $ess(\mathbf{U} \cup \{\tau_1\}, \mathcal{S}) \subseteq ess(\mathbf{U} \cup \{\tau_2\}, \mathcal{S})$ *if, and only if,* $core(\mathbf{U} \cup \{\tau_2\}, \mathcal{S}) \longrightarrow core(\mathbf{U} \cup \{\tau_1\}, \mathcal{S})$*.*

Proof. First, assume that $ess(\mathbf{U} \cup \{\tau_1\}, \mathcal{S}) \subseteq ess(\mathbf{U} \cup \{\tau_2\}, \mathcal{S})$. Hence, by Definition 12, $inst(core(\mathbf{U} \cup \{\tau_1\}, \mathcal{S}), \mathcal{S}) \subseteq inst(core(\mathbf{U} \cup \{\tau_2\}, \mathcal{S}), \mathcal{S})$. Now, since $\mathbf{U} \cup \{\tau_1\} \subseteq inst(core(\mathbf{U} \cup \{\tau_1\}, \mathcal{S}), \mathcal{S})$, then $\mathbf{U} \cup \{\tau_1\} \subseteq inst(core(\mathbf{U} \cup \{\tau_2\}, \mathcal{S}), \mathcal{S})$. Therefore, $core(\mathbf{U} \cup \{\tau_2\}, \mathcal{S})$ explains $\mathbf{U} \cup \{\tau_1\}$, while $core(\mathbf{U} \cup \{\tau_1\}, \mathcal{S})$ characterizes $\mathbf{U} \cup \{\tau_1\}$. Thus, by Corollary 3, $core(\mathbf{U} \cup \{\tau_2\}, \mathcal{S}) \longrightarrow core(\mathbf{U} \cup \{\tau_1\}, \mathcal{S})$.

Now, assume that $core(\mathbf{U} \cup \{\tau_2\}, \mathcal{S}) \longrightarrow core(\mathbf{U} \cup \{\tau_1\}, \mathcal{S})$. Hence, exploiting Proposition 5, we know that $inst(core(\mathbf{U} \cup \{\tau_1\}, \mathcal{S}), \mathcal{S})$ is a subset of $inst(core(\mathbf{U} \cup \{\tau_2\}, \mathcal{S}), \mathcal{S})$, and the two may also coincide. Thus, we can conclude that $ess(\mathbf{U} \cup \{\tau_1\}, \mathcal{S}) \subseteq ess(\mathbf{U} \cup \{\tau_2\}, \mathcal{S})$. □

5.2 The Expansion Graph

We are now ready to introduce the key notion of this section.

Definition 13. *The* expansion graph *of* $\mathbf{U}$ *according to* $\mathcal{S}$*, denoted by* $eg(\mathbf{U}, \mathcal{S})$*, is the triple* (N, A, δ) *where:*

(*i*) N *is the set* $\{[core(\mathbf{U} \cup \{\tau\}, \mathcal{S})] : \tau \in \mathbf{T}\}$*;*

(*ii*) A *is a set collecting each pair* $([\varphi_1], [\varphi_2])$ *of distinct nodes of* N *such that* $\varphi_2 \longrightarrow \varphi_1$ *and there is no other* $[\varphi_3] \in N$ *such that* $\varphi_2 \longrightarrow \varphi_3$ *and* $\varphi_3 \longrightarrow \varphi_1$*; and*

(*iii*) δ *is a function mapping each* $[\varphi] \in N$ *to the set* $inst(\varphi, \mathcal{S}) \setminus \{\tau \in inst(\varphi', \mathcal{S}) : ([\varphi'], [\varphi]) \in A\}$*, named the set of all* direct instances *of each formula in* $[\varphi]$*, and denoted by* $dinst(\varphi, \mathcal{S})$*.* ■

Note that, as the elements of the domain of the function δ are equivalence classes, to be sure that δ is well-defined, we need to show that given two distinct formulas in $[\varphi]$, say φ_1 and φ_2, then $inst(\varphi_1, \mathcal{S}) \setminus \{\tau \in inst(\varphi', \mathcal{S}) : ([\varphi'], [\varphi_1]) \in A\} = inst(\varphi_2, \mathcal{S}) \setminus \{\tau \in inst(\varphi', \mathcal{S}) : ([\varphi'], [\varphi_2]) \in A\}$. Intuitively, this is a consequence of the fact that $\varphi_2 \in [\varphi_1]$ implies $\varphi_1 \longleftrightarrow \varphi_2$, thus, it can be applied Proposition 5. More formally,

Proposition 13. *Function* δ *in* $eg(\mathbf{U}, \mathcal{S})$ *is well-defined.*

Proof. By Definition 13, δ maps every $[\varphi] \in N$ to the set $inst(\varphi, \mathcal{S}) \setminus \{\tau \in inst(\varphi', \mathcal{S}) : ([\varphi'], [\varphi]) \in A\}$. What we need to show is that this result does not depend on the chosen representative. To this aim, consider two distinct formulas φ_1 and φ_2 both in $[\varphi]$ and two distinct formulas φ_3 and φ_4 both in $[\varphi']$. By definition we get $\varphi_1 \simeq \varphi_2 \simeq \varphi$ and $\varphi_3 \simeq \varphi_4 \simeq \varphi'$, hence by Proposition 5 $inst(\varphi_1, \mathcal{S}) = inst(\varphi_2, \mathcal{S}) = inst(\varphi, \mathcal{S})$ and $inst(\varphi_3, \mathcal{S}) = inst(\varphi_4, \mathcal{S}) = inst(\varphi', \mathcal{S})$. This shows that the set $inst(\varphi, \mathcal{S}) \setminus \{\tau \in inst(\varphi', \mathcal{S}) : ([\varphi'], [\varphi]) \in A\}$ does not depend on the particular choice of the representative for the equivalence classes $[\varphi]$ and $[\varphi']$. Hence the set $inst(\varphi, \mathcal{S}) \setminus \{\tau \in inst(\varphi', \mathcal{S}) : ([\varphi'], [\varphi]) \in A\}$ is well defined. Therefore δ is well-defined, that is what we wanted to show. □

Example 26. *Consider again our running example. The expansion graph* $eg(\mathbf{U}_0, \mathcal{S}_0)$ *is depicted in Figure 1.2, together with the shapes of all core characterizations. Node-labels are of the form* $\varphi \mapsto \delta([\varphi])$*, where* φ *is one representative element of* $[\varphi]$*. In particular, the set of nodes is* $N = \{[\bar{\varphi}_a], [\bar{\varphi}_b], [\bar{\varphi}_c], [\bar{\varphi}_d], [\bar{\varphi}_e], [\bar{\varphi}_f]\}$*, where*

$$
\begin{array}{rcl}
[\bar{\varphi}_a] & = & [core(\mathbf{U}_0, \mathcal{S}_0)], \\
[\bar{\varphi}_b] & = & [core(\mathbf{U}_0 \cup \{\langle \mathsf{Pacific\ Park} \rangle\}, \mathcal{S}_0)], \\
[\bar{\varphi}_c] & = & [core(\mathbf{U}_0 \cup \{\langle \mathsf{Gardaland} \rangle\}, \mathcal{S}_0)], \\
[\bar{\varphi}_d] & = & [core(\mathbf{U}_0 \cup \{\langle \mathsf{Prater} \rangle\}, \mathcal{S}_0)] = [core(\mathbf{U}_0 \cup \{\langle \mathsf{Leolandia} \rangle\}, \mathcal{S}_0)], \\
[\bar{\varphi}_e] & = & [core(\mathbf{U}_0 \cup \{\langle \mathsf{tp} \rangle\}, \mathcal{S}_0)], \\
[\bar{\varphi}_f] & = & [core(\mathbf{U}_0 \cup \{\langle \mathsf{Italy} \rangle\}, \mathcal{S}_0)] = [core(\mathbf{U}_0 \cup \{\langle \mathsf{Austria} \rangle\}, \mathcal{S}_0)] = \\
 & = & [core(\mathbf{U}_0 \cup \{\langle \mathsf{Florida} \rangle\}, \mathcal{S}_0)] = [core(\mathbf{U}_0 \cup \{\langle \mathsf{California} \rangle\}, \mathcal{S}_0)] = \\
 & = & [core(\mathbf{U}_0 \cup \{\langle \mathsf{ap} \rangle\}, \mathcal{S}_0)] = [core(\mathbf{U}_0 \cup \{\langle \mathsf{US} \rangle\}, \mathcal{S}_0)];
\end{array}
$$

the set of edges is $A = \{([\bar{\varphi}_a], [\bar{\varphi}_b]),\ ([\bar{\varphi}_a], [\bar{\varphi}_c]),\ ([\bar{\varphi}_b], [\bar{\varphi}_d]),\ ([\bar{\varphi}_c], [\bar{\varphi}_d]),\ ([\bar{\varphi}_d], [\bar{\varphi}_e]),\ ([\bar{\varphi}_e],\ [\bar{\varphi}_f])\}$; *and, finally, the function* δ *maps*

$$
\begin{array}{rcll}
\bar{\varphi}_a & \mapsto & inst(\bar{\varphi}_a, \mathcal{S}_0) \setminus \{\tau \in inst(\varphi', \mathcal{S}_0) : ([\varphi'], [\bar{\varphi}_a]) \in A\} & = \\
 & = & \mathbf{U}_0, & \\
\bar{\varphi}_b & \mapsto & inst(\bar{\varphi}_b, \mathcal{S}_0) \setminus \{\tau \in inst(\varphi', \mathcal{S}_0) : ([\varphi'], [\bar{\varphi}_b]) \in A\} & = \\
 & = & \{\langle \mathsf{Pacific\ Park} \rangle\}, & \\
\bar{\varphi}_c & \mapsto & inst(\bar{\varphi}_c, \mathcal{S}_0) \setminus \{\tau \in inst(\varphi', \mathcal{S}_0) : ([\varphi'], [\bar{\varphi}_c]) \in A\} & = \\
 & = & \{\langle \mathsf{Gardaland} \rangle\}, & \\
\bar{\varphi}_d & \mapsto & inst(\bar{\varphi}_d, \mathcal{S}_0) \setminus \{\tau \in inst(\varphi', \mathcal{S}_0) : ([\varphi'], [\bar{\varphi}_d]) \in A\} & = \\
 & = & \{\langle \mathsf{Prater} \rangle, \langle \mathsf{Leolandia} \rangle\}, & \\
\bar{\varphi}_e & \mapsto & inst(\bar{\varphi}_e, \mathcal{S}_0) \setminus \{\tau \in inst(\varphi', \mathcal{S}_0) : ([\varphi'], [\bar{\varphi}_e]) \in A\} & = \\
 & = & \{\langle \mathsf{tp} \rangle\}, & \\
\bar{\varphi}_f & \mapsto & inst(\bar{\varphi}_f, \mathcal{S}_0) \setminus \{\tau \in inst(\varphi', \mathcal{S}_0) : ([\varphi'], [\bar{\varphi}_f]) \in A\} & = \\
 & = & \{\langle \mathsf{Italy} \rangle, \langle \mathsf{California} \rangle, \langle \mathsf{US} \rangle, \langle \mathsf{Florida} \rangle, \langle \mathsf{Austria} \rangle, \langle \mathsf{ap} \rangle\}. &
\end{array}
$$

Indeed,

$$
\begin{array}{rcl}
inst(\bar{\varphi}_a, \mathcal{S}_0) & = & \{\langle \mathsf{Discovery\ Cove} \rangle, \langle \mathsf{Epcot} \rangle\}, \\
inst(\bar{\varphi}_b, \mathcal{S}_0) & = & \{\langle \mathsf{Discovery\ Cove} \rangle, \langle \mathsf{Epcot} \rangle, \langle \mathsf{Pacific\ Park} \rangle\}, \\
inst(\bar{\varphi}_c, \mathcal{S}_0) & = & \{\langle \mathsf{Discovery\ Cove} \rangle, \langle \mathsf{Epcot} \rangle, \langle \mathsf{Gardaland} \rangle\}, \\
inst(\bar{\varphi}_d, \mathcal{S}_0) & = & \{\langle \mathsf{Discovery\ Cove} \rangle, \langle \mathsf{Epcot} \rangle, \langle \mathsf{Pacific\ Park} \rangle, \langle \mathsf{Gardaland} \rangle, \\
 & & \langle \mathsf{Prater} \rangle, \langle \mathsf{Leolandia} \rangle\}, \\
inst(\bar{\varphi}_e, \mathcal{S}_0) & = & \{\langle \mathsf{Discovery\ Cove} \rangle, \langle \mathsf{Epcot} \rangle, \langle \mathsf{Pacific\ Park} \rangle, \langle \mathsf{Gardaland} \rangle, \\
 & & \langle \mathsf{Prater} \rangle, \langle \mathsf{Leolandia} \rangle, \langle \mathsf{tp} \rangle\}, \\
inst(\bar{\varphi}_f, \mathcal{S}_0) & = & \{\langle \xi \rangle : \xi \textit{ is an entity}\}.
\end{array}
$$

Since $ess(\mathbf{U}_0, \mathcal{S}_0) = \mathbf{U}_0$, *the direct instances of* $\bar{\varphi}_a$ *are exactly the tuples in* $\mathbf{U}_0$. ∎

We conclude this section by showing some useful properties of the expansion graph.

Theorem 3. *Let $eg(\mathbf{U}, \mathcal{S})$ be the expansion graph of $\mathbf{U}$ according to $\mathcal{S}$. Then, the following statements do hold:*

(1) *$eg(\mathbf{U}, \mathcal{S})$ is directed and acyclic;*

(2) *the sets of direct instances forms a partition of $\mathbf{T}$;*

(3) *$[core(\mathbf{U}, \mathcal{S})]$ is the only source node;*

(4) *$\delta([core(\mathbf{U}, \mathcal{S})]) = ess(\mathbf{U}, \mathcal{S})$.*

Proof. Let $eg(\mathbf{U}, \mathcal{S})$ be the expansion graph of $\mathbf{U}$ according to $\mathcal{S}$.

(1) First, note that the expansion graph $eg(\mathbf{U}, \mathcal{S})$ is directed by construction. Concerning the acyclicity, by contradiction, we observe that the existence of a cycle in $eg(\mathbf{U}, \mathcal{S})$ would require all formulas in the nodes involved to belong to the same equivalence class. Indeed, assume that there exist distinct nodes $[\varphi_1]$, $[\varphi_2]$, ..., $[\varphi_n]$ such that $([\varphi_{i+1}], [\varphi_i]) \in A$, for each $i = 1, \ldots, n-1$, and also $([\varphi_1], [\varphi_n]) \in A$. Then, $\varphi_1 \longrightarrow \varphi_2$, ..., $\varphi_{n-1} \longrightarrow \varphi_n$, and $\varphi_n \longrightarrow \varphi_1$. Therefore, by C-homomorphism composition, $\varphi_1 \longleftrightarrow \varphi_2$, ..., $\varphi_{n-1} \longleftrightarrow \varphi_n$, and $\varphi_n \longleftrightarrow \varphi_1$. Hence, in particular, $\varphi_i \in [\varphi_1]$, for each $i = 1, \ldots, n$, that is a contradiction.

(2) To prove that the sets of direct instances forms a partition of $\mathbf{T}$, we need to show that $(i) \ \bigcup_{[\varphi] \in N} dinst(\varphi, \mathcal{S}) = \mathbf{T}$ and (ii) $dinst(\varphi_1, \mathcal{S}) \cap dinst(\varphi_2, \mathcal{S}) = \emptyset$, for each pair of distinct nodes $[\varphi_1]$, $[\varphi_2] \in N$. Concerning (i), let $\tau \in \mathbf{T}$. By construction, there is $[\varphi] \in N$ such that $[\varphi] = [core(\mathbf{U} \cup \{\tau\}, \mathcal{S})]$. Hence, φ characterizes $\mathbf{U} \cup \{\tau\}$ and $\tau \in inst(\varphi, \mathcal{S})$. First, we claim that $\tau \in \delta([\varphi])$. By contradiction, assume that there is some $([\varphi'], [\varphi]) \in A$ such that $\varphi \longrightarrow \varphi'$, $\varphi' \not\longrightarrow \varphi$, and $\tau \in inst(\varphi', \mathcal{S})$. Thus, φ' explains $\mathbf{U} \cup \{\tau\}$. But $\varphi' \not\longrightarrow \varphi$ violates Corollary 3. Concerning (ii), assume that there are two distinct nodes $[\varphi]$ and $[\varphi']$ in N such that $\tau \in \delta([\varphi])$ and $\tau \in \delta([\varphi'])$. Hence, $\tau \in inst(\varphi, \mathcal{S})$ and $\tau \in inst(\varphi', \mathcal{S})$ and, thus, both φ and φ' explain $\mathbf{U} \cup \{\tau\}$. In particular, by Corollary 3, $\varphi \longrightarrow \varphi'$ and $\varphi' \longrightarrow \varphi$. Hence $\varphi \longleftrightarrow \varphi'$, which is a contradiction.

(3) Now, we show that $[core(\mathbf{U}, \mathcal{S})]$ is the only source node. First, note that $[core(\mathbf{U}, \mathcal{S})]$ is a node of the expansion graph, as it is equal to $[core(\mathbf{U} \cup \{\tau\}, \mathcal{S})]$, for each $\tau \in \mathbf{U}$. Then, to show that $[core(\mathbf{U}, \mathcal{S})]$ is a source node, we assume, by contradiction, that there exists another node $[core(\mathbf{U} \cup \{\tau\}, \mathcal{S})]$, for some $\tau \in \mathbf{T}$, and this node is such that $([core(\mathbf{U} \cup \{\tau\}, \mathcal{S})], [core(\mathbf{U}, \mathcal{S})]) \in A$. Hence, by definition of A, we have that $core(\mathbf{U}, \mathcal{S}) \longrightarrow core(\mathbf{U} \cup \{\tau\}, \mathcal{S})$. Therefore, by Proposition 12, $ess(\mathbf{U} \cup \{\tau\}, \mathcal{S}) \subseteq ess(\mathbf{U}, \mathcal{S})$. Now, since $\mathbf{U} \cup \{\tau\} \subseteq ess(\mathbf{U} \cup \{\tau\}, \mathcal{S})$,

then $\mathbf{U} \cup \{\tau\} \subseteq ess(\mathbf{U}, \mathcal{S})$. Then, by Proposition 11, $core(\mathbf{U}, \mathcal{S})$ characterizes $\mathbf{U} \cup \{\tau\}$. Since $core(\mathbf{U}, \mathcal{S})$ characterizes $\mathbf{U} \cup \{\tau\}$ and $\mathbf{U} \subseteq \mathbf{U} \cup \{\tau\}$, we get that $core(\mathbf{U}, \mathcal{S})$ explains $\mathbf{U}$. Therefore, by Corollary 3, $core(\mathbf{U} \cup \{\tau\}, \mathcal{S}) \longrightarrow core(\mathbf{U}, \mathcal{S})$. Hence, $core(\mathbf{U}, \mathcal{S}) \longleftrightarrow core(\mathbf{U} \cup \{\tau\}, \mathcal{S})$, which is a contradiction. Finally, to show that $[core(\mathbf{U}, \mathcal{S})]$ is the only source node, assume, by contradiction, that there is another source node, say $[core(\mathbf{U} \cup \{\tau\}, \mathcal{S})]$, for some $\tau \in \mathbf{T}$. Hence, $core(\mathbf{U} \cup \{\tau\}, \mathcal{S})$ explains $\mathbf{U}$, and $core(\mathbf{U}, \mathcal{S})$ characterizes $\mathbf{U}$. Therefore, by Corollary 3, $core(\mathbf{U} \cup \{\tau\}, \mathcal{S}) \longrightarrow core(\mathbf{U}, \mathcal{S})$, which is a contradiction.

(4) Finally, to prove that $\delta([core(\mathbf{U}, \mathcal{S})]) = ess(\mathbf{U}, \mathcal{S})$, we observe that $\delta([core(\mathbf{U}, \mathcal{S})]) = inst([core(\mathbf{U}, \mathcal{S})], \mathcal{S})$, since $[core(\mathbf{U}, \mathcal{S})]$ is a source. Hence, the claim holds as, by definition, $inst([core(\mathbf{U}, \mathcal{S})], \mathcal{S}) = ess(\mathbf{U}, \mathcal{S})$. □

5.3 Applicability to Semantic Similarity and Entity Set Expansion

Expansion graphs do provide a natural bridge from nexus characterizations to semantic similarity [33] and entity set expansion [63].

First, the analysis of $eg(\mathbf{U}, \mathcal{S})$ provides a way to rank (tuple of) entities according to their degree of similarity with respect to the initial unit $\mathbf{U}$. For instance, according to Example 1, Figure 1.2, and Example 26, one may see that:

(*i*) since $core(\mathbf{U}_0 \cup \{\langle \mathsf{Leolandia} \rangle\}, \mathcal{S}_0) \simeq \bar{\varphi}_d$ is a generalization of the formula $core(\mathbf{U}_0 \cup \{\langle \mathsf{Gardaland} \rangle\}, \mathcal{S}_0) \simeq \bar{\varphi}_c$ (namely, $\bar{\varphi}_d \longrightarrow \bar{\varphi}_c$ and $\bar{\varphi}_c \not\longrightarrow \bar{\varphi}_d$), then the nexus of similarity that $\langle \mathsf{Gardaland} \rangle$ has with $\mathbf{U}_0$ are higher than those that $\langle \mathsf{Leolandia} \rangle$ has with $\mathbf{U}_0$, and this implies that $\langle \mathsf{Gardaland} \rangle$ is more similar to the entities of $\mathbf{U}_0$ than $\langle \mathsf{Leolandia} \rangle$;

(*ii*) since we know that $core(\mathbf{U}_0 \cup \{\langle \mathsf{Prater} \rangle\}, \mathcal{S}_0) \simeq \bar{\varphi}_d$ is equivalent to $core(\mathbf{U}_0 \cup \{\langle \mathsf{Leolandia} \rangle\}, \mathcal{S}_0) \simeq \bar{\varphi}_d$, then the nexus of similarity that $\langle \mathsf{Prater} \rangle$ has with $\mathbf{U}_0$ coincide with those that $\langle \mathsf{Leolandia} \rangle$ has with $\mathbf{U}_0$, and this means that $\langle \mathsf{Prater} \rangle$ is as similar as $\langle \mathsf{Leolandia} \rangle$ to the entities of $\mathbf{U}_0$;

(*iii*) since $core(\mathbf{U}_0 \cup \{\langle \mathsf{Pacific\_Park} \rangle\}, \mathcal{S}_0) \simeq \bar{\varphi}_b$ is incomparable to the formula $core(\mathbf{U}_0 \cup \{\langle \mathsf{Gardaland} \rangle\}, \mathcal{S}_0) \simeq \bar{\varphi}_c$ (namely, $\bar{\varphi}_b \not\longrightarrow \bar{\varphi}_c$ and $\bar{\varphi}_c \not\longrightarrow \bar{\varphi}_b$), then the nexus of similarity that $\langle \mathsf{Pacific\_Park} \rangle$ has with $\mathbf{U}_0$ are incomparable to those that $\langle \mathsf{Gardaland} \rangle$ has with $\mathbf{U}_0$.

Second, the navigation of $eg(\mathbf{U}, \mathcal{S})$ from its source node $[core(\mathbf{U}, \mathcal{S})]$ provides possible "expansion plans" of the initial unit $\mathbf{U}$. For instance, by considering again Example 1 and Figure 1.2, one may note that $\mathbf{U}_0 = ess(\mathbf{U}_0, \mathcal{S}_0)$ has two possible expansion plans in $eg(\mathbf{U}_0, \mathcal{S}_0)$, depending on whether $\langle \mathsf{Gardaland} \rangle$ comes either before or after $\langle \mathsf{Pacific\_Park} \rangle$, as they are incomparable:

$$\begin{gathered}\mathbf{U}_0 = \{\langle \mathsf{Discovery\_Cove} \rangle, \langle \mathsf{Epcot} \rangle\} \ \subset \ \mathbf{U}_1 = \\ = \mathbf{U}_0 \ \cup \ \{\langle \mathsf{Pacific\_Park} \rangle\} \ \subset \ \mathbf{U}_2 = \\ = \mathbf{U}_1 \ \cup \ \{\langle \mathsf{Gardaland} \rangle, \langle \mathsf{Prater} \rangle, \langle \mathsf{Leolandia} \rangle\} \ \subset \ \ldots\end{gathered}$$

$$\begin{gathered}\mathbf{U}_0 = \{\langle \mathsf{Discovery\_Cove} \rangle, \langle \mathsf{Epcot} \rangle\} \ \subset \ \mathbf{U}'_1 = \\ = \mathbf{U}_0 \ \cup \ \{\langle \mathsf{Gardaland} \rangle\} \ \subset \ \mathbf{U}'_2 = \\ = \mathbf{U}'_1 \ \cup \ \{\langle \mathsf{Pacific\_Park} \rangle, \langle \mathsf{Prater} \rangle, \langle \mathsf{Leolandia} \rangle\} \ \subset \ \ldots\end{gathered}$$

This confirms that a unit may admit multiple meaningful linear expansions. Choosing one or another may depend on specific user preferences or some other information being part of the specific application scenario and that are not encoded in the given knowledge base.

Finally, existing techniques designed to work over taxonomies (see, for example, [17]) can be used with an expansion graph to get also similarity scores. In particular, one can focus on those techniques that preserve the similarity order induced by the comparable nodes of the expansion graph. Moreover, such techniques can provide useful extra information about incomparable nodes. For example, if there is an agreement that $\langle \mathsf{Guardaland} \rangle$ is more similar to $\mathbf{U}_0$ than $\langle \mathsf{Pacific\_Park} \rangle$, then one could opt for the second expansion plan above rather than the first one.

Chapter 6

Computational Aspects

In the upcoming chapter, we will methodically introduce a series of issues, including both decision problems and research problems, that are closely linked to our formal framework. Understanding these tasks is crucial for a deeper insight into the nexus of similarity between tuples of entities. At the same time, we aim to define the limits of these problems' tractability through thorough examination under three distinct theoretical assumptions.

6.1 Reasoning Tasks

The primary objective of this section is to meticulously introduce and comprehensively analyze, from a computational standpoint, a selection of captivating tasks that are strictly related to our framework. Our initial step will be to delineate some key definitions. These will serve as building blocks, helping us easily introduce the aforementioned computational tasks that have a two-fold connection to our framework. Let ω denote the maximum

Problem	**Input**	**Reasoning Task**
CAN	$\mathcal{S}, \mathbf{U}$	compute $can(\mathbf{U}, \mathcal{S})$
CORE	$\mathcal{S}, \mathbf{U}$	compute $core(\mathbf{U}, \mathcal{S})$
ESS	$\mathcal{S}, \mathbf{U}, \tau$	does $\tau \in ess(\mathbf{U}, \mathcal{S})$ hold?
PREC	$\mathcal{S}, \mathbf{U}, \tau, \tau'$	does $\tau \prec^{\mathcal{S}}_{\mathbf{U}} \tau'$ hold?
SIM	$\mathcal{S}, \mathbf{U}, \tau, \tau'$	does $\tau \sim^{\mathcal{S}}_{\mathbf{U}} \tau'$ hold?
INC	$\mathcal{S}, \mathbf{U}, \tau, \tau'$	does $\tau \| ^{\mathcal{S}}_{\mathbf{U}} \tau'$ hold?
GAD1	$\mathcal{S}, \mathbf{U}, \tau, \tau'$	does $\tau \in ess(\mathbf{U} \cup \{\tau'\}, \mathcal{S})$ hold?
GAD2	$\mathcal{S}, \mathbf{U}, \tau, \tau'$	does $\tau' \in ess(\mathbf{U} \cup \{\tau\}, \mathcal{S})$ hold?

Table 6.1: Key reasoning tasks; everywhere $\{\tau, \tau'\} \cap \mathbf{U} = \emptyset$.

arity inside D. Given two n-ary tuples τ and τ' over $\mathbb{D}_D$, we say that:

(i) the nexus of similarity that τ has with $\mathbf{U}$ are *higher than* those that τ' has with $\mathbf{U}$, written $\tau \prec^{\mathcal{S}}_{\mathbf{U}} \tau'$, if $ess(\mathbf{U} \cup \{\tau\}, \mathcal{S}) \subset ess(\mathbf{U} \cup \{\tau'\}, \mathcal{S})$;

(ii) the nexus of similarity that τ has with $\mathbf{U}$ *coincide with* those that τ' has with $\mathbf{U}$ (with respect to $\mathcal{S}$), written $\tau \sim^{\mathcal{S}}_{\mathbf{U}} \tau'$, if $ess(\mathbf{U} \cup \{\tau\}, \mathcal{S}) = ess(\mathbf{U} \cup \{\tau'\}, \mathcal{S})$; and

(iii) the nexus of similarity that τ has with $\mathbf{U}$ are *incomparable to* those that τ' has with $\mathbf{U}$, (with respect to $\mathcal{S}$), written $\tau \| ^{\mathcal{S}}_{\mathbf{U}} \tau'$, if $\tau \prec^{\mathcal{S}}_{\mathbf{U}} \tau'$, $\tau' \prec^{\mathcal{S}}_{\mathbf{U}} \tau$, and $\tau \sim^{\mathcal{S}}_{\mathbf{U}} \tau'$ do not hold.

We are now ready to define the computational problems reported in Table 6.1, where we specify their inputs and the associated reasoning tasks. In particular, we consider the tasks CAN and CORE of computing $can(\mathbf{U}, \mathcal{S})$ and $core(\mathbf{U}, \mathcal{S})$, respectively. Also, we consider the task ESS of checking whether a tuple $\tau \in \mathbf{T}$ belongs to $ess(\mathbf{U}, \mathcal{S})$. This task serves as the building block for expansion graphs. Tasks PREC, SIM and INC allow to compare tuples within an expansion graph. Specifically, task PREC of checking whether tuples $\tau, \tau' \in \mathbf{T}$ are such that the nexus of similarity that τ has with $\mathbf{U}$ are *higher than* those that τ' has with $\mathbf{U}$ (with respect to $\mathcal{S}$), task SIM of checking whether tuples $\tau, \tau' \in \mathbf{T}$ are such that the nexus of similarity that τ has with $\mathbf{U}$ *coincide with* those that τ' has with $\mathbf{U}$ (with respect to $\mathcal{S}$) and task INC of checking whether tuples $\tau, \tau' \in \mathbf{T}$ are such that the nexus of similarity that τ has with $\mathbf{U}$ are *incomparable to* those that τ' has with $\mathbf{U}$ (with respect to $\mathcal{S}$). Finally, tasks GAD1 and GAD2 are useful gadgets to simplify the analysis for some of the previous tasks.

In what follows, we chart the tractability frontier of these tasks under three computational assumptions:

1. *broad*, where the maximum arity ω of predicates in K is bounded by some fixed integer (ω is typically "small" and even limited to two in cases $\mathcal{S}$ is derived from a KG) and both $\varsigma(K, \tau)$ and $ent(K)$ are polynomial-time computable;

2. *medium*, where, in addition, $m = |\mathbf{U}|$ is bounded (this value is typically two in semantic similarity, and typically referred to as "small" in entity set expansion); and

3. *handy*, where, in addition, $|\mathbb{D}_{\varsigma(K,\tau)}| \in \mathcal{O}(\sqrt[m+1]{\log_2 |\mathbb{D}_{ent(K)}|})$ (the number of constants of a summary is typically referred to as "small" in entity summarization).

Problem	Broad	Medium	Handy
CORE	in $F\mathsf{EXP}^{\mathsf{NP}} \setminus F\mathsf{P}^{\mathsf{PH}}$	in $F\mathsf{P}^{\mathsf{NP}} \setminus F\mathsf{P}^{\dagger}$	in $F\mathsf{P}$
CAN	in $F\mathsf{PSPACE} \setminus F\mathsf{P}^{\mathsf{PH}}$	in $F\mathsf{P}$	in $F\mathsf{P}$
ESS	NEXP-complete	NP-complete	in P
PREC	DEXP-complete	DP-complete	in P
SIM	NEXP-complete	NP-complete	in P
INC	coNEXP-complete	coNP-complete	in P

Table 6.2: Computational complexity under the three different theoretical assumptions. The lower bound marked by † holds under the assumption that $\mathsf{NP} \neq \mathsf{coNP}$

Note that the asymptotic bound for the handy case comes out of the following observation.

Proposition 14. *In the handy case, the total number of all possible* C*-homomorphism from* $atm(can(\mathbf{U}, \mathcal{S}))$ *to* $\varsigma(\tau)$ *for any possible* τ *is at most polynomial with respect to* $\mathbb{D}_{ent(K)}$.

Proof. Let y be the number of variables in $can(\mathbf{U}, \mathcal{S})$. Then, we know that y is at most z^m, where $z = \mathsf{max}(|\mathbb{D}_{\varsigma(\tau,\mathcal{S})}| \ : \ \tau \in \mathbf{U})$ and m is the cardinality of the unit $\mathbf{U}$. Hence, the total number of possible mapping from this set of variables to $\mathbb{D}_{\varsigma(\tau,\mathcal{S})}$ for a fixed τ is z^{z^m}. Since we want this quantity to be at most polynomial with respect to $\mathbb{D}_{ent(K)}$, we need to satisfy the following inequality $z^{z^m} \leq |\mathbb{D}_{ent(K)}|^{\bar{k}}$ for some $\bar{k} \in \mathbb{N}$. Taking logarithms on both side (noticing that argument are at least equal to 1 and not less) we get $\log_2 z^{z^m} = z^m \log_2 z \leq \log_2 |\mathbb{D}_{ent(K)}|^{\bar{k}} = \bar{k} \log_2 |\mathbb{D}_{ent(K)}|$. Now, since we know that $\log_2 z < z$, we can increase the quantity $z^m \log_2 z$ to z^{m+1}, hence imposing the following, more stringent, condition $z^{m+1} \leq \bar{k} \log_2 |\mathbb{D}_{ent(K)}|$. From there, we derive $z \leq \sqrt[m+1]{\bar{k} \log_2 |\mathbb{D}_{ent(K)}|} = \sqrt[m+1]{\bar{k}} \sqrt[m+1]{\log_2 |\mathbb{D}_{ent(K)}|}$. We have just shown that $z = |\mathbb{D}_{\varsigma(K,\tau)}| \in \mathcal{O}(\sqrt[m+1]{\log_2 |\mathbb{D}_{ent(K)}|})$ implies that the total number of all possible C-homomorphism from $atm(can(\mathbf{U}, \mathcal{S}))$ to $\varsigma(\tau)$ for any possible τ is at most polynomial with respect to $\mathbb{D}_{ent(K)}$, that is what we wanted to show. □

6.2 Computing Characterizations

Having seen in detail how to build procedurally $can(\mathbf{U}, \mathcal{S})$ while reading Chapter 4, we can now easily analyze its structure.

In particular, we observe that its computational complexity in terms of size exhibits different patterns depending on the scenario. In the broad case, despite the upper bound ω being fixed, the complexity grows exponentially. This should not surprise us at all since from the construction it is clear that the exponentiality depends on the number of tuples present in our unit.

There is therefore a natural remedy that can be applied; in the medium case where both ω and m are bounded, the size of $can(\mathbf{U}, \mathcal{S})$ becomes polynomial. Although this property already seems to be favorable, there is a small hidden element that leaves us thinking: we have no guarantee that the number of variables present in the formula is less than polynomial, and this, as is typical, involves a possible exponential number of C-homomorphisms.

It is therefore necessary to run further for cover but also in this case the solution is not far away. From this premise, i.e., the desire to check the possible number of C-homomorphisms that may exist between the atoms of the formula we built and the various summaries on which we make our evaluations, the need for the handy case arises. Indeed, in the latter case, although the magnitude of the formula may remain polynomial, the constraint on the total number of variables, deriving from a bound to the summaries domain, guarantees us the possibility of keeping at bay (potentially by enumerating them all one by one) all possible C-homomorphisms that start from the atoms of the formula and arrive in the various summaries, ergo, as we will see later, the handy case does not guarantee simple polynomial in the size but also polynomial in the answers, making us able to effectively evaluate when an element is or is not in the ess of a unit.

Before validating the elements just introduced, it is good to have a clear formulation of how to calculate not only $can(\mathbf{U}, \mathcal{S})$ but also $core(\mathbf{U}, \mathcal{S})$. So, below we will present and discuss some useful algorithms that will help us along our way.

CAN

This section will be entirely dedicated to the calculation of a canonical characterization given an initial unit. We will introduce ad hoc algorithms of which we will evaluate the computational cost as well as other results that will prove useful in the following. We start by Algorithm 1 for computing $can(\mathbf{U}, \mathcal{S})$. According to the construction in Chapter 4: line 1 constructs the set Fr; line 2 prints the free variables of $can(\mathbf{U}, \mathcal{S})$ and n atoms that always belong to it; lines 3–4 build $P \cup C$; line 5 checks, for each $\beta \in P \cup C$, whether $\mu(\beta)$ has a (direct or indirect) connection with some atom containing a free variable; and, if so, line 6 appends $\mu(\beta)$ to the output. In particular, $\mathsf{NearCon}$ is implemented by Algorithm 2, which mimics nondeterministic

Algorithm 1: BuildCan$(\mathbf{U}, \mathcal{S})$

Input: $\mathcal{S} = (K, \varsigma)$ and $\mathbf{U} = \{\tau_1, ..., \tau_m\}$.

$d_{\mathbf{s}_1}, ..., d_{\mathbf{s}_n} := \tau_1 \otimes ... \otimes \tau_m$
print $x_{\mathbf{s}_1}, ..., x_{\mathbf{s}_n} \leftarrow \top(x_{\mathbf{s}_1}) \wedge ... \wedge \top(x_{\mathbf{s}_n})$
for $\alpha \in \varsigma(\tau_1) \otimes ... \otimes \varsigma(\tau_m)$ **do**
    for $\beta \in \{\alpha\} \cup clones(\alpha)$ **do**
        if NearCon$(\beta, \mathbf{U}, \mathcal{S})$ == Yes **then**
            print $\wedge\, \mu(\beta)$

Algorithm 2: NearCon$(\beta, \mathbf{U}, \mathcal{S})$

Input: $\mathcal{S} = (K, \varsigma)$, $\mathbf{U} = \{\tau_1, ..., \tau_m\}$ and an atom β.

guess $d_{\mathbf{s}} \in \tau_1 \otimes ... \otimes \tau_m$
if $x_{\mathbf{s}} \in \mathbb{D}_{\{\mu(\beta)\}}$ **then**
    accept this branch
guess $\alpha \in \varsigma(\tau_1) \otimes ... \otimes \varsigma(\tau_m)$
if $\mathbb{D}_{\{\mu(\alpha)\}} \cap \mathbb{D}_{\{\mu(\beta)\}} \neq \emptyset$ **then**
    if NearCon$(\alpha, \mathcal{S}, \mathbf{U})$ == Yes **then**
        accept this branch
reject this branch

graph reachability: given an atom β, it checks whether $\mu(\beta)$ contains a free variable; if so, then it accepts, else it guesses an "adjacent" atom α and, recursively, repeat the former check. For notational convenience hereafter, for each $j \in \mathbb{N}^+$, $[j]$ is a shorthand for $\{1, .., j\}$.

Theorem 4. *Problem* CAN *belongs to* $F\mathsf{PSPACE} \setminus F\mathsf{P}^{\mathsf{PH}}$ *in the broad case and to* $F\mathsf{P}$ *both in the medium and handy case.*

Proof. We begin by considering the upper bounds, specifically focusing on the broad case of the problem.

In the general scenario, Algorithm 1 exhibits a computational complexity of $F\mathsf{PSPACE}$ while utilizing $\mathsf{NearCon}$ as an oracle within $\mathsf{NPSPACE}$. This complexity arises from its approach of sequentially considering each constituent atom within the exponentially large structure associated with the direct product, rather than materializing the entire structure at once.

Since we know that $F\mathsf{PSPACE}^{\mathsf{NPSPACE}} = F\mathsf{PSPACE}$, the result follows directly.

Moving on to the remaining cases, due to the boundedness of m, the following statements hold:

1. Algorithm 1 runs in $F\mathsf{P}$.
2. $\mathsf{NearCon}$ runs in NL.

The result then follows since we know that $F\mathsf{P}^{\mathsf{NL}} = F\mathsf{P}$.

Turning to the lower bounds, to demonstrate the unavoidability of the exponentiality of $can(\mathbf{U}, \mathcal{S})$ in the broad case, we establish the following result.

Let c be the constant 2^{ω}. It holds that

$$|can(\mathbf{U}, \mathcal{S})| \leq c \cdot \prod_{i \in [m]} |\varsigma(\tau_i)|.$$

In particular, there exists a family $\{(\mathbf{U}_{\bar{m}}, \mathcal{S}_{\bar{m}})\}_{\bar{m}>1}$, where $\mathbf{U}_{\bar{m}} = \{\bar{\tau}_1, ..., \bar{\tau}_{\bar{m}}\}$ is a unary unit, and $\mathcal{S}_{\bar{m}} = (K_{\bar{m}}, \bar{\varsigma})$ is an SKB with $K_{\bar{m}} = (D_{\bar{m}}, O_{\bar{m}})$, $|\bar{\varsigma}(K_{\bar{m}}, \bar{\tau}_j)| > 2$, $j \in \{1, ..., m\}$, such that

$$|can(\mathbf{U}_{\bar{m}}, \mathcal{S}_{\bar{m}})| = 2^{1-\bar{m}} \cdot \prod_{i \in [\bar{m}]} |\bar{\varsigma}(\bar{\tau}_i)|.$$

This result implies that such an object cannot be constructed in $F\mathsf{P}^{\mathsf{PH}}$, concluding the proof.

To establish the upper bound, we first show that $|can(\mathbf{U}, \mathcal{S})| \leq c \cdot \prod_{i \in [m]} |\varsigma(\tau_i)|$.

Since, by construction, $|can(\mathbf{U},\mathcal{S})| \leq |\Phi(\mathbf{U},\mathcal{S})|$ holds, demonstrating $|\Phi(\mathbf{U},\mathcal{S})| \leq c \cdot \prod_{i\in[m]} |\varsigma(\tau_i)|$ suffices.

We observe this by examining the construction of the formula $\Phi(\mathbf{U},\mathcal{S})$. Constructed as a connected formula, $\Phi(\mathbf{U},\mathcal{S})$ is defined as follows:

$$\Phi(\mathbf{U},\mathcal{S}) = x_{\mathbf{s}_1} \leftarrow \bigwedge_{p(t_1,...t_k)\in P} p(\mu(t_1),...,\mu(t_k)) =$$

$$= \bigwedge_{r(t_1,t_2)\in P} r(\mu(t_1),\mu(t_2)) \ \cup \ \bigwedge_{t\in\mathbb{D}_P} \top(\mu(t)).$$

Since $|P| = |\Gamma_N|$ by construction, we have that $|P|$, and thus the size of $can(\mathbf{U},\mathcal{S})$, is exponential with respect to the parameter m.

For the lower bounds, we explicitly construct the family $\{(\mathbf{U}_{\bar{m}},\mathcal{S}_{\bar{m}})\}_{\bar{m}>0}$ and prove that $|can(\mathbf{U}_{\bar{m}},\mathcal{S}_{\bar{m}})| = 2^{1-\bar{m}} \cdot \prod_{i\in[\bar{m}]} |\bar{\varsigma}(\bar{\tau}_i)|$.

To achieve this, let us proceed step by step:

(i) *Construction of the Family* $\{(\mathbf{U}_{\bar{m}},\mathcal{S}_{\bar{m}})\}_{\bar{m}>0}$. Let $\mathbf{pr}_{\bar{m}}$ be the $\bar{m}$-th prime number and

$$\Gamma_i = \{\mathsf{r}(\mathsf{c}^i_1,\mathsf{c}^i_2),...,\mathsf{r}(\mathsf{c}^i_{\mathbf{pr}_i-1},\mathsf{c}^i_{\mathbf{pr}_i}),\mathsf{r}(\mathsf{c}^i_{\mathbf{pr}_i},\mathsf{c}^i_1)\} \ \cup \ \{\top(\mathsf{c}^i_j) : j \in [\mathbf{pr}_i]\}$$

for $i \in \mathbb{N}$.

Define $D_{\bar{m}} = \ \cup \ _{i\in[\bar{m}]}\Gamma_i$, $O_{\bar{m}} = \emptyset$, $K_{\bar{m}} = (D_{\bar{m}},O_{\bar{m}})$ and $\bar{\varsigma}$ such that $\bar{\varsigma}(K_{\bar{m}},\bar{\tau}_i) = \Gamma_i$ with $\bar{\tau}_i = \langle \mathsf{c}^i_1\rangle$ for $i \in [\bar{m}]$; finally, consider $\mathbf{U}_{\bar{m}} = \cup_{i\in[\bar{m}]}\{\langle \mathsf{c}^i_1\rangle\}$.

(ii) *Verification of* $|can(\mathbf{U}_{\bar{m}},\mathcal{S}_{\bar{m}})| = 2^{1-\bar{m}} \cdot \prod_{i\in[\bar{m}]} |\bar{\varsigma}(\bar{\tau}_i)|$. For the construction of $can(\mathbf{U}_{\bar{m}},\mathcal{S}_{\bar{m}})$, we define the dataset $P = \bar{\varsigma}(\bar{\tau}_1) \otimes ... \otimes \bar{\varsigma}(\bar{\tau}_{\bar{m}}) = \bar{\varsigma}(\langle c^1_1\rangle) \otimes ... \otimes \bar{\varsigma}(\langle c^{\bar{m}}_1\rangle)$.

Let $N = \prod_{i\in[\bar{m}]}[\mathbf{pr}_i]$. For every $n \in [N]$, let $\mathsf{rem}(n,i)$ be the remainder of the division $n/\mathbf{pr}_i$. We define $f_{\bar{m}}(n) = c^1_{s(1,n)},...,c^{\bar{m}}_{s(\bar{m},n)}$, where

$$s(i,n) = \begin{cases} \mathsf{rem}(n,i) & \text{if } \mathsf{rem}(n,i) \neq 0 \\ \mathbf{pr}_i & \text{otherwise} \end{cases}$$

Given these definitions we get that

$$P = \{\top(d_{f_{\bar{m}}(n)}) \mid n \in [N]\} \ \cup \ \{r(d_{f_{\bar{m}}(N)},d_{f_{\bar{m}}(1)})\} \ \cup$$

$$\cup \ \{r(d_{f_{\bar{m}}(n)},d_{f_{\bar{m}}(n+1)}) \mid n \in [N-1]\}.$$

Thus, P itself is a connected structure. By construction, no element from $d_s \in \mathbb{D}_P$ is such that $|\mathbb{D}_{\mathbf{s}}| = 1$. Hence, $Ge = \{d_{\mathbf{s}} \in \mathbb{D}_P : |\mathbb{D}_{\mathbf{s}}| = 1\} = \emptyset$.

We construct μ as the mapping $\{d_{\mathbf{s}} \mapsto g(d_{\mathbf{s}}) : d_{\mathbf{s}} \in \mathbb{D}_P\}$, where $g(d_{\mathbf{s}})$ is the mapping $\{d_{\mathbf{s}} \mapsto x_{\mathbf{s}} : d_{\mathbf{s}} \in Fr\} \ \cup \ \{d_{\mathbf{s}} \mapsto y_{\mathbf{s}} : d_{\mathbf{s}} \notin Fr\}$, since $Ge = \emptyset$.

Now, we can define

$$\Phi(\mathbf{U}_{\bar{m}}, \mathcal{S}_{\bar{m}}) = x_{\mathbf{s}_1} \leftarrow \bigwedge_{p(t_1,...t_k)\in P} p(\mu(t_1), ..., \mu(t_k)) =$$

$$= \bigwedge_{r(t_1,t_2)\in P} r(\mu(t_1), \mu(t_2)) \ \cup \ \bigwedge_{t\in\mathbb{D}_P} \top(\mu(t)).$$

Since the formula $\Phi(\mathbf{U}_{\bar{m}}, \mathcal{S}_{\bar{m}})$ just constructed is connected, it coincides with $can(\mathbf{U}_{\bar{m}}, \mathcal{S}_{\bar{m}})$.

Moreover, since $|P| = |\Gamma_N|$, $|P|$ and hence the size of $can(\mathbf{U}_{\bar{m}}, \mathcal{S}_{\bar{m}})$, is exponential concerning the parameter $\bar{m}$.

This concludes the proof, establishing both upper and lower bounds for the given problem.

□

Following the previous proof we can derive a corollary that will be useful in the following.

Corollary 6. *There exists a family* $\{(\mathbf{U}_{\bar{m}}, \mathcal{S}_{\bar{m}})\}_{\bar{m}>0}$, *where* $\mathbf{U}_{\bar{m}} = \{\bar{\tau}_1, ..., \bar{\tau}_{\bar{m}}\}$ *is a unary unit and* $\mathcal{S}_{\bar{m}} = (K_{\bar{m}}, \bar{\varsigma})$ *is an SKB with* $\bar{\varsigma}(K_{\bar{m}}, \bar{\tau}_m) \subset \bar{\varsigma}(K_{\bar{m}+1}, \bar{\tau}_{m+1})$, *such that*

$$|core(\mathbf{U}_{\bar{m}}, \mathcal{S}_{\bar{m}})| = |can(\mathbf{U}_{\bar{m}}, \mathcal{S}_{\bar{m}})| = 2^{1-\bar{m}} \cdot \prod_{i\in[\bar{m}]} |\bar{\varsigma}(\bar{\tau}_i)|.$$

Proof. The proof of this corollary follows from the proof of Theorem 4 noting that the constructed

$$can(\mathbf{U}_{\bar{m}}, \mathcal{S}_{\bar{m}}) = x_{\mathbf{s}_1} \leftarrow \bigwedge_{r(t_1,t_2)\in P} r(\mu(t_1), \mu(t_2)) \ \cup \ \bigwedge_{t\in\mathbb{D}_P} \top(\mu(t))$$

is always a core of itself. This last statement follows since a core of a single (finite) cycle is always isomorphic to the cycle itself and the part $\cup_{r(t_1,t_2)\in P} r(\mu(t_1), \mu(t_2))$ forms a cycle of cardinality $N = \prod_{i\in[\bar{m}]}[\mathbf{pr}_i]$, where $\mathbf{pr}_i$ we are representing the i-th prime number as before. On the other hand since we are not able to delete any of the atoms in this cycle we are neither able to delete any of the atoms in $\cup_{t\in\mathbb{D}_P} \top(\mu(t))$, otherwise we would get an absurd since we would have been able to delete also an element coming from the cycle. □

However, the canonical type characterizations, as will be evident in the light of the reading of the results taken from Chapter 7, although they possess excellent properties from the computational point of view, they (often) sin in terms of readability for man, in the next section we will therefore have a precise focus on how to overcome this apparent drawback.

CORE

In this section, we will focus exclusively on how to compute a core characterization from a given initial unit. We will present specific algorithms for this task and analyze their computational cost as well as other useful results for the next sections. We continue our analysis by presenting Algorithm 3 for computing $core(\mathbf{U}, \mathcal{S})$. This time: line 1 constructs $\varphi = can(\mathbf{U}, \mathcal{S})$ and collects its atoms in A; line 2 enumerates each $\alpha \in A$; line 3 builds φ' from φ by removing α; line 4 checks whether $\varphi \longrightarrow \varphi'$; if so, line 5 copies φ' in φ; finally, after removing all redundant atoms, line 6 prints φ.

Algorithm 3: $\mathsf{BuildCore}(\mathbf{U}, \mathcal{S})$

Input: $\mathcal{S} = (K, \varsigma)$ and $\mathbf{U} = \{\tau_1, ..., \tau_m\}$.

```
1 φ := BuildCan(U, S)    and    A := atm(φ)
2 for α ∈ A do
3 |   φ' := remove(α, φ)
4 |   if φ ⟶ φ' then
5 |   |   φ := φ'
6 print φ
```

Theorem 5. CORE *is in* $F\mathsf{EXP}^{\mathsf{NP}} \setminus F\mathsf{P}^{\mathsf{PH}}$ *in the broad case, in* $F\mathsf{P}^{\mathsf{NP}}$ *in the medium case and in* $F\mathsf{P}$ *in the handy case. Unless* $\mathsf{NP} = \mathsf{coNP}$, CORE $\notin F\mathsf{P}$ *in the medium case.*

Proof. We begin by analyzing the upperbounds of Algorithm 3 in the broad case.

In particular, we notice that in this case the Algorithm runs in $F\mathsf{EXP}$ using an oracle in NP for checking whether $\varphi \longrightarrow \varphi'$ holds.

Next, we examine the medium case for Algorithm 3.

In this case, Algorithm 3 runs in $F\mathsf{P}$ using again an oracle in NP always for checking whether $\varphi \longrightarrow \varphi'$ holds.

In the handy case, everything becomes polynomial as the size of φ and the number of its variables are bounded. About the lower bounds, we need to consider both the broad and the medium case.

For what concerns the broad case, Corollary 6 leads us to conclude that in general such an object, being exponential, cannot be constructed in $F\mathsf{P}^{\mathsf{PH}}$.

For what concerns the medium case instead, let CORE-IDENTIFICATION be the DP-complete problem reported in [28]: *given a pair* (S', S) *of finite structures such that* $S' \subseteq S$, *is* S' *isomorphic to* $core(S)$?

Let $\mathbb{NT}$ be all constant-free formulas in N$\mathbb{C}$F closed under $\top$, and NT-CORE be the analogous of CORE-IDENTIFICATION for $\mathbb{NT}$ formulas, in other terms, given a pair (φ', φ) of $\mathbb{NT}$ formulas such that $atm(\varphi') \subseteq atm(\varphi)$, is φ' isomorphic to $core(\varphi)$?

Indeed, NT-CORE remains DP-hard. To show this last statement, first notice that exploiting the proof of Theorem 4.2 in [28], we can consider, as input for CORE-IDENTIFICATION, two undirected graphs.

Now consider the following composition of mappings $(G', G) \mapsto (\varphi', \varphi) \mapsto (\varphi''', \varphi'')$, where G, G' are two undirected graphs, $\varphi, \varphi', \varphi''$ and φ''' are all constants-free formulas. Moreover, $G' \subseteq G$, $atm(\varphi') \subseteq atm(\varphi)$, $atm(\varphi''') \subseteq atm(\varphi'')$, φ and φ' are over a single binary relation, call it r, and both φ''' and φ'' are $\mathbb{NT}$ formulas.

To construct φ starting from G (respectively φ' starting from G'), you simply consider two not open formulas such that $G = atm(\varphi)$ and $G' = atm(\varphi')$.

To construct φ'' starting from φ (respectively φ''' starting from φ'), consider the fresh relation name s and without loss of generality say that $x \notin \mathbb{D}_{atm(\varphi)}$, then we can construct

$$\varphi'' = x \leftarrow \bigwedge_{\alpha \in atm(\varphi)} \alpha \wedge \bigwedge_{y \in \mathbb{D}(\varphi)} s(x, y) \wedge \bigwedge_{z \in \mathbb{D}(\varphi) \cup \{x\}} \top(z)$$

and

$$\varphi''' = x \leftarrow \bigwedge_{\alpha \in atm(\varphi')} \alpha \wedge \bigwedge_{y \in \mathbb{D}(\varphi')} s(x, y) \wedge \bigwedge_{z \in \mathbb{D}(\varphi') \cup \{x\}} \top(z)$$

.

It is straightforward to notice that G' is the core of G if and only if φ' is the core of φ, if and only if φ''' is the core of φ''.

That G' is the core of G if and only if φ' is the core of φ is obvious.

For the other, on one hand say that φ' is the core of φ and say by contradiction that φ''' is not the core of φ'', since by construction $\varphi''' \mapsto \varphi''$ the only way possible is that it exists $\tilde{\varphi} \in \mathbb{NT}$ such that $atm(\tilde{\varphi}) \subset atm(\varphi'')$ and $\tilde{\varphi}$ is the core of φ''' (that coincides by definition with the core of φ'').

Let α be an atom in $atm(\varphi'') \setminus atm(\tilde{\varphi})$, such an atom has to exist in order to have $atm(\tilde{\varphi}) \subset atm(\varphi'')$. By construction, the only 3 possibilities for this atom are (i) $\alpha = r(z, z')$ for some $z, z' \in \mathbb{D}_{atm(\varphi'')}$, (ii) $\alpha = s(x, z)$ for some $z \in \mathbb{D}_{atm(\varphi'')}$ and (iii) $\alpha = \top(z)$ for some $z \in \mathbb{D}_{atm(\varphi'')}$.

Now we will prove that for any of these 3 possibilities, we will end up with a contradiction. First, say that α is of the form $r(z, z')$ for some $z, z' \in \mathbb{D}_{atm(\varphi'')}$, then by definition it exists a homomorphism h from $atm(\varphi'')$ to $atm(\varphi'')$ such that at least one of the following two does hold: either

$h(z) \neq z$ or $h(z \neq z'$. Say that $h(z) \neq z$, the other implication can be solved by using a similar argument. Then we could use the same homomorphism h on h on the variables present in φ' and thus obtain that $\tilde{\varphi'}$ exists such that $atm(\tilde{\varphi'}) \subset atm(\varphi')$, contradicting the assumption that φ' is a core. All other remaining possibilities give rise to the same contradiction following the same construction.

On the other hand, say that φ' is not the core of φ, then obviously φ''' is not the core of φ''. Now we show that for each $\varphi \in \mathbb{NT}$, it is possible to construct in $F\mathsf{P}$ a pair $(\mathcal{S}, \mathbf{U})$ such that both $\varphi \simeq can(\mathbf{U}, \mathcal{S})$ and $core(\varphi) \simeq core(\mathbf{U}, \mathcal{S})$ hold. In order to do so, starting from a k-ary $\varphi \in \mathbb{NT}$, without loss of generality consider its free variables to be $x_1, ..., x_k$, and consider the following knowledge base $K = (D, \emptyset)$, where the dataset $D = \{p(cz_1, ..., cz_n) \ : \ p(z_1, ..., z_n) \in atm(\varphi)\} \cup \{p(\mathsf{alias}, ..., \mathsf{alias}) \ : \ p(z_1, ..., z_n) \in atm(\varphi)\}$, $\mathbf{U} = \{\langle cx_1, ..., cx_k\rangle, \langle \mathsf{alias}, ..., \mathsf{alias}\rangle\}$. As the last thing, we have to define our summary selector ς as the following function $\varsigma(K, \tilde{\tau})$ that returns D whatever the input tuple $\tilde{\tau}$ will be. The really important thing to note in this construction is that the initial database domain is bipartite. In particular, there exist no $p(z_1, ..., z_n) \in D$ such that $z_i = \mathsf{alias}$ for some $i \in [n]$ and then there exist $j \in [n]$ such that $z_i \neq z_j$. This important property tells us a priori an important thing about the construction of the canonical explanation of the unit we built, in particular the set Ge restricted to any connected part of P that contains d_s with $s = cx_i, \mathsf{alias}$ with $i \in [k]$ will always be empty and this tells us we will have no constants at all in the set of atoms of $can(\mathbf{U}, \mathcal{S})$; moreover, $atm(can(\mathbf{U}, \mathcal{S}))$ will be isomorphic to $atm(\varphi)$ by construction. To note this, the simplest way is to see that each constant of the direct product that is constructed using the algorithm provided in Section 4.3 will have the form $d_{c,\mathsf{alias}}$ for some constant c present in the union of the connected parts of the dataset containing the constants $cx_1, ..., cx_k$ but by construction this is isomorphic to the starting formula.

We can now reduce NT-CORE to CORE in the medium case. From a pair (φ', φ) of formulas in $\mathbb{NT}$, do: (i) construct in $F\mathsf{P}$ a pair $(\mathcal{S}, \mathbf{U})$ such that $\varphi \simeq can(\mathbf{U}, \mathcal{S})$; (ii) construct $core(\mathbf{U}, \mathcal{S})$; and (iii) check in NP whether $\varphi' \simeq core(\mathbf{U}, \mathcal{S})$. If CORE were in $F\mathsf{P}$ in the medium case, then step (ii) would also be in $F\mathsf{P}$ and, hence, NT-CORE would be in NP, which is impossible unless $\mathsf{NP} = \mathsf{coNP}$.

□

Building characterizations, however, is not all that we set out to do. We are in fact also interested in understanding when a given tuple shares more or less nexus of similarity with a given unit compared to another tuple or if perhaps these nexus are incomparable with each other. What is

all this useful for? One possible immediate application is Recommendation Systems.

Imagine a user on a streaming platform searching exclusively for films within the drama genre. It is reasonable to assume that such films share a higher nexus of similarity with other dramas due to common thematic elements. For instance, if a user watches *The Shawshank Redemption*, the system might suggest other profound dramas like *Forrest Gump* or *The Green Mile*.

However, consider a comedy film such as *Dumb and Dumber*. Despite being a successful film, it shares less nexus of similarity with the drama genre due to its comedic nature and distinct narrative style.

Thus, a Recommendation System with access to the order induced starting from the degree of nexus of similarity shared by the elements would intelligently suggest drama films over comedies to a user who has previously searched for predominantly dramatic films.

6.3 Navigating Expansion Graphs

In this section, we delve into essential notations that form the backbone of our reductions. To ensure clarity, we provide intuitive explanations for each notation, often accompanied by illustrative examples. Additionally, we draw upon established results from existing literature, which underpin the complexity analyses presented here. Throughout the section, we formally define various decision tasks, each accompanied by dedicated reference sections. These references elaborate on the problem's significance and practical applications.

Let us establish some fundamental notation. Consider the partition of set C into $\{\mathsf{C}_f, \mathsf{C}_\ell\}$, where C_f represents flat constants (e.g., a, a_1, a_2) and $\mathsf{C}_\ell = \{c^s : c \in \mathsf{C}_f \wedge s > 1\}$ represents lifted constants.

We further categorize C_f into *flat* and *lifted* constants. The *flat* constants are denoted as $\mathsf{C}_{re} = \{\mathsf{alias}\} \cup \{\mathsf{a}_i, \mathsf{b}_i : i > 0\}$, and the remaining flat constants form the *input* set $\mathsf{C}_{in} = \mathsf{C}_f \setminus \mathsf{C}_{re}$. Similarly, predicates are divided into *reserved* (P_{re}) and *input* (P_{in}) sets, ensuring specific symbols are exclusively used in our constructions.

The concept of *focus* and *twins* is fundamental. For any $k > 0$ and $a \in \mathsf{C}_f$, let $fc(C)$ be the set of focus predicates for elements in C, and $tw(C, k)$ be the set generating *twins* of constants in C. These twins create $k - 1$ new elements for each constant in C.

To better grasp this, consider an example that illustrates the construction and significance of these twins.

Example 27. *Let* $D = \{r(1,2)\}$*, clearly* $\mathbb{D}_D = \{1,2\}$*. We want to know what* $fc(\mathbb{D}_D)$ *and* $tw(D,2)$ *look like. By their own definition we have that* $fc(\mathbb{D}_D) = \{\mathsf{focus}(1), \mathsf{focus}(2)\}$ *and* $tw(D,2) = \{\mathsf{twin}(1,1^2)\,, \mathsf{twin}(2,2^2), \top(1^2), \top(2^2)\}$*.* ■

Starting from a unary tuple $\tau = \langle a \rangle$, and an integer $k \geq 1$, let $\tau^k = \langle a, a^2, ..., a^k \rangle$. From a unary unit $\mathbf{U} = \{\tau_1, ..., \tau_m\}$, let $\mathbf{U}^k$ be $\{\tau_1^k, ..., \tau_m^k\}$.

Example 28. *Consider the following unit* $\mathbf{U} = \{\langle 1 \rangle, \langle 2 \rangle\}$*, then we can construct* $\mathbf{U}^3$ *that is equal to* $\{\langle 1 \rangle^3, \langle 2 \rangle^3\} = \{\langle 1, 1^2, 1^3 \rangle, \langle 2, 2^2, 2^3 \rangle\}$*.* ■

From a dataset D, let $double(D,a)$ be the dataset defined as $\{p(c_1, ..., c_{|p|}) : p(b_1, ..., b_{|p|}) \in D \wedge \text{each } c_i \in g(b_i)\}$, where $g(a) = \{a, \mathsf{alias}\}$, and $g(b_i) = \{b_i\}$ whenever $b_i \neq a$. Please note that, as the following example will explain, what we will obtain, in general, will not be a completely detached copy of the initial dataset. Instead, it will be a new structure that remains connected to the original.

Example 29. *If we consider a dataset* D *to be equal to* $\{\mathsf{r}(1,2)\}$*, then we have that* $double(D,2)$ *is equal to* $\{\mathsf{r}(1,2), \mathsf{r}(1,\mathsf{alias})\}$*.* ■

From $\tau \in \mathsf{C}^h$, let $off(\tau, a) \in \mathsf{C}_f^h$ be the tuple obtained from τ by replacing each c^s with c and alias with a. Intuitively, this operation might be mistaken as the inverse of the previously defined operation.

Although it will be used in that way later on, it is important to note that in general, this would not be guaranteed if the constant alias were not part of the reserved constants to which we do not have access as potential inputs. The next example will clarify this assertion.

Example 30. *Let* $D = \{r(1,2), r(3,a), r(1^4, 2^5\}$*, then* $off(\tau, 4)$ *applied to all (unary) tuples constructed starting from* $\mathbb{D}_D$ *will result in the set* $\{\langle 1 \rangle, \langle 2 \rangle, \langle 3 \rangle, \langle 4 \rangle\}$*. In particular, this operation works on tuples and not datasets.* ■

From $\tau \in \mathsf{C}^k$, let $off(\tau, a) \in \mathsf{C}_f^k$ be the tuple obtained from τ by replacing each c^s with c and alias with a. Considering that non-flat constants and alias are not part of our possible inputs, this operation ensures that, when and if activated, it will simply be another way for us to regain access to an object related to a potential input. Intuitively, for example, we could use this object to retrieve a specific summary whose trends we were already familiar with. The following example will help the reader in better understanding this concept.

Example 31. *Let* $\tau = \langle \mathsf{alias}, 1^3, 2, b^2 \rangle$*, clearly is not a possible input tuples for any of our reductions since it uses* alias*,* 1^3 *and* b^2*. What about off*(τ, b)*? The final result is* $\langle b, 1, 2, b \rangle$ *that is a possible input for our reductions.* ∎

From two tuples $\langle \mathbf{t}_1 \rangle$ and $\langle \mathbf{t}_2 \rangle$ of any arities, let $\langle \mathbf{t}_1 \rangle \circ \langle \mathbf{t}_2 \rangle$ be $\langle \mathbf{t}_1, \mathbf{t}_2 \rangle$. From two units $\mathbf{U}_1$ and $\mathbf{U}_2$, let $\mathbf{U}_1 \times \mathbf{U}_2$ be the unit $\{\tau_1 \circ \tau_2 : \tau_1 \in \mathbf{U}_1 \wedge \tau_2 \in \mathbf{U}_2\}$.

The operation defined above is nothing more than the classical *Cartesian product*. In the remainder of this thesis the operation will be always carried out in cases where $\mathbb{D}_{\mathbf{U}_1} \cap \mathbb{D}_{\mathbf{U}_2} = \emptyset$.

Example 32. *Accordingly let* $\mathbf{U}_1 = \{\langle a, b \rangle, \langle b, c \rangle\}$ *and* $\mathbf{U}_2 = \{\langle 1 \rangle\}$*, then we have that* $\mathbf{U}_1 \times \mathbf{U}_2 = \{\tau_1 \circ \tau_2 : \tau_1 \in \mathbf{U}_1 \wedge \tau_2 \in \mathbf{U}_2\} = \{\langle a, b, 1 \rangle, \langle b, c, 1 \rangle\}$. ∎

The subsequent concepts derived from the literature necessitate remembrance.

An (*undirected*) graph $\mathcal{G}$ is a ordered pair (V, E) where $V = \{v_1, ..., v_n\}$, for some fixed $n \in \mathbb{N}$, is the set of its vertices while $E \subseteq V \times V$ is the set of its (undirected) edges, that is $(i, j) \in E \implies (j, i) \in E$.

Given a graph $\mathcal{G} = (V, E)$ two distinct vertices, $i, j \in V$, are said to be *adjacents* if there exist $e \in E$ such that $e = (i, j) \vee e = (j, i)$.

A *k-coloring* (or *k-vertex coloring*), for some fixed $k \in \mathbb{N}$, of a graph $\mathcal{G} = (V, E)$ is a mapping $h : V \mapsto [k]$ such that no two adjacents vertices are mapped to the same element.

A graph $\mathcal{G} = (V, E)$ is *complete* if, however taken two distinct vertices $i, j \in V$, they are adjacent.

Given a graph $\mathcal{G}$ the *k-colorability* problem asks if it is possible to find a k-coloring for $\mathcal{G}$.

The following definition is not typical, the tool we are presenting will prove to be extremely useful in the following.

Definition 14. *Given a graph* $\mathcal{G} = (V, E)$ *the graph* $\mathcal{G}^a = (V^a, E^a)$*, where* $V^a = V \cup \{a\}$ *(with "a" fresh vertex) and* $E^a = E \cup \{(a, v) : v \in V\} \cup \{(v, a) : v \in V\}$*, is called* complement to a *of* $\mathcal{G}$*.*

Example 33. *Let us consider the following datataset* $D = \{\mathsf{r}(1,2), \mathsf{r}(1,3), \mathsf{r}(2,3), \mathsf{r}(3,1), \mathsf{r}(3,2), \mathsf{r}(2,1)\}$ *that can be seen as a relational representation of a graph* $\mathcal{G}$*. Then* $\mathcal{G}^a$ *is represented by* $D^a = D \cup \{\mathsf{r}(a,1), \mathsf{r}(a,2), \mathsf{r}(a,3), \mathsf{r}(1,a), \mathsf{r}(2,a), \mathsf{r}(3,a)\}$*.* ∎

We now recall the following important result from the literature.

Remark 1. *It is well known [45] that the problem of* k-Col *for* $k \geq 3$ *is NP-complete for undirected graphs.*

It is also well known that a graph $\mathcal{G} = (V, E)$ is n-colourable if and only if it exists a homomorphism from $\mathcal{G}$ to K_n, where K_n is the complete graph of order n.

Lemma 1. *A graph $\mathcal{G} = (V, E)$ is k-colourable if and only if the complement to a of $\mathcal{G}$ is $(k+1)$-colourable.*

Proof of Lemma 1. $\implies$ Say that $\mathcal{G}$ is k-colourable, this means that it exists $h : V \mapsto [k]$ such that no two adjacents vertices are mapped to the same element. From h we can construct $\bar{h} : V^a \mapsto [k+1]$ in the following fashion:

$$\bar{h}(v) = \begin{cases} h(v), & \forall v \in V \\ k+1, & \text{otherwise} \end{cases}$$

$\impliedby$ Say that $\mathcal{G}^a$ is (k+1)-colourable, this means that it exists $\bar{h} : V^a \mapsto [k+1]$ such that no two adjacents vertices are mapped to the same element. Without loss of generality we may say that $\bar{h}(a) = k + 1$. From $\bar{h}$ we can construct $h : V \mapsto [k]$ simply by putting

$$h(v) = \bar{h}(v), \forall v \in V$$

In fact, since a is adjacent to any other vertex in V^a by construction if $\bar{h}$ is a proper (k+1)-colouring then no other element of V^a is mapped to k+1, but this means that every element of V is mapped to $[k]$ by $\bar{h}$ and thus h is a proper k-colouring of $\mathcal{G}$. □

Now everything is in place to start our computational trip. For any decision problem π let π_k be its version having an input unit $\mathbf{U}$ of arity k. Then we derive the following.

ESS

Theorem 6. ESS *is* NEXP*-complete,* NP*-complete and in* P *in the broad, medium and handy case, respectively. In particular, Lower Bounds hold already for* ESS_k *with $k > 0$, a selective $\mathcal{S}$ with an empty ontology and a unit $|\mathbf{U}| > 1$.*

Proof. For what concerns the upper bounds, we can make use of Algorithm 4. Let us consider the three cases separately. In the broad case, line 1 runs in exponential time, lines 2 run in nondeterministic exponential time, and lines 3-4 run in exponential time. In the medium case, line 1 runs in polynomial time, lines 2 run in nondeterministic polynomial time, and lines 3-4 run in polynomial time. For the handy case, instead of guessing

a mapping as we do in Algorithm 4, due to Proposition 14, we can simply enumerate all of them. Since they are polynomially many, the resulting algorithm will run in polynomial time.

For what concerns the lower bounds, we will give two distinct constructions, one for the broad case and one for the medium one. Let PHP-SB be the NEXP-complete problem: *given a sequence* $\mathbf{s} = I_1, ..., I_{m+1}$ *of instances over the binary relation* br *with* $\mathbb{D}_{I_i} \cap \mathbb{D}_{I_{j\neq i}} = \emptyset$*, is there any homomorphism from* $I_1 \otimes ... \otimes I_m$ *to* I_{m+1}*?* Let 3-COL be the NP-complete problem: *given a graph* $G = (V, E)$*, is there a map* $\lambda : V \to \{\mathtt{r}, \mathtt{g}, \mathtt{b}\}$ *such that* $\{u, v\} \in V$ *implies* $\lambda(u) \neq \lambda(v)$*?*

In the broad case, for each $k > 0$, by adapting a technique presented in [77], we show that PHP-SB $\leq$ ESS$_k$. To do so let $\mathbf{s}$ be an input for PHP-SB and consider the following map: $\mathbf{s} \mapsto (\mathcal{S}, \mathbf{U}^k, \langle \mathsf{a}_{m+1} \rangle^k)$, where:

- $\mathcal{S} = (K, \varsigma)$,
- $K = (D, O)$,
- $D = D_1 \ \cup \ ... \ \cup \ D_4$,
- $I_{m+2} = \{\mathsf{br}(\mathsf{a}_{m+2}, \mathsf{a}_{m+2})\}$,
- $D_1 = I_1 \ \cup \ ... \ \cup \ I_{m+2}$,
- $D_2 = \{\top(c) : c \in \mathbb{D}_{D_1}\}$,
- $D_3 = \{\top(\mathsf{a}_i), \mathsf{br}(\mathsf{a}_i, c) : c \in \mathbb{D}(I_i)\}_{i \in [m+2]}$,
- $D_4 = tw(\{\mathsf{a}_1, ..., \mathsf{a}_{m+2}\}, k)$,
- $O = \emptyset$,
- ς always selects $D^\top$,
- $\mathbf{U} = \{\langle \mathsf{a}_{m+2} \rangle\} \ \cup \ \{\langle \mathsf{a}_1 \rangle, ..., \langle \mathsf{a}_m \rangle\}$.

The reason why we are adding the element $\langle a_{m+2} \rangle$ to our unit is linked to the fact that from the point of view of possible homomorphisms it does not change anything, however, even if the starting m had been equal to one, it guarantees us that there can be no use of any constant within the formulas that we are going to build. It is, in fact, easy to check that there is a homomorphism from $I_1 \otimes ... \otimes I_m$ to I_{m+1} if, and only if, there is a homomorphism from $I_1 \otimes ... \otimes I_m \otimes I_{m+2}$ to I_{m+1}. First, let us notice that

in our context, given a characterization for $\mathbf{U}$, call it φ, then we can easily construct a characterization for $\mathbf{U}^k$ of the form

$$x_1, ..., x_k \leftarrow \bigwedge_{s\in[2...k]} \top(x_s), \mathsf{twin}_s(x_1, x_s) \wedge \bigwedge_{\alpha\in atm(\varphi)} \alpha.$$

Vice versa, say that you have a characterization φ' for $\mathbf{U}^k$, then by renaming every occurrence of the free variables other than the first one, we will end up with a characterization for $\mathbf{U}$. According to this, we can simply focus on the unary case and then add this detail later. Let S denote $I_1 \otimes ... \otimes I_m \otimes I_{m+2}$. In one direction, if $S \mapsto I_{m+1}$ then, by homomorphism preservation, any conjunctive formula that explains the elements of $\mathbf{U}$ will also explain $\langle a_{m+1}\rangle$. On the other hand, if $S \not\mapsto I_{m+1}$, then we can construct a formula $\varphi \in \mathrm{N\mathbb{C}F}$ that explains $\mathbf{U}$ but not $\langle a_{m+1}\rangle$. To this aim, first consider φ_S as the formula

$$\bigwedge_{\mathsf{br}(t_1,t_2)\in S} \mathsf{br}(y_{t_1}, y_{t_2}).$$

Then take φ as

$$x \leftarrow \varphi_S \bigwedge_{y\in\mathbb{D}_{\varphi_S}} \mathsf{br}(x, y).$$

By construction, φ explains all the elements of $\mathbf{U}$ but does not explain $\langle a_{m+1}\rangle$. In order to see this, on the one hand consider, for each $i \in [m+2] \setminus \{m+1\}$, h_i as the projection onto the i-th component. Obviously, each of these is a homomorphism from $I_1 \otimes ... \otimes I_m \otimes I_{m+2} \mapsto I_i$, and in particular, this shows that φ_S is true when evaluated on every I_i with $i \in [m+2] \setminus \{m+1\}$, implying $\langle a_i\rangle \in inst(\varphi, \mathcal{S}) \forall i \in [m+2] \setminus \{m+1\}$. On the other hand, suppose by contradiction that there exists a homomorphism h $atm(\varphi)$ to $\varsigma(a_{m+1})$ such that $\langle h(x)\rangle = \langle a_{m+1}\rangle$, then the same homomorphism h limited to variables other than x would show the existence of a homomorphism h' from $I_1 \otimes ... \otimes I_m \otimes I_{m+2}$ to I_{m+1}, and this is absurd. What remains to be shown is that the new summary selector can be built without making complexity leaps and that its execution time turns out to be polynomial with respect to the input received, but this is straightforward since the summary selector has always to return the entire dataset in input.

We have just completed the proof concerning the broad case, and we can, therefore, proceed to the next case. We will now show that for each $k > 0$, in the medium case, 3-COL $\leq$ ESS$_k$ via the following map: $G \mapsto (\mathcal{S}, \mathbf{U}^k, \langle \mathsf{b}_1\rangle^k)$, where:

- $\mathcal{S} = (K, \varsigma)$,

- $K = (D, O)$,
- $D = D_1 \ \cup \ ... \ \cup \ D_5$,
- $D_1 = \{\mathsf{arc}(u_i, v_i), \mathsf{arc}(v_i, u_i), : \{u, v\} \in E \wedge i \in [2]\}$,
- $D_2 = \{\top(c) : c \in \mathbb{D}_{D_1}\}$,
- $D_3 = \{\top(\mathsf{b}_i), \mathsf{arc}(\mathsf{b}_i, \mathsf{b}_z) : i, z \in [4], i \neq z\}$,
- $D_4 = \{\top(\mathsf{a}_i), \mathsf{arc}(\mathsf{a}_i, v_i), \mathsf{arc}(v_i, \mathsf{a}_i) : v \in V \wedge i \in [2]\}$,
- $D_5 = tw(\mathbb{D}_{D_3} \ \cup \ \mathbb{D}_{\mathbf{U}}, k)$,
- $O = \emptyset$,
- ς always selects D,
- $\mathbf{U} = \{\langle \mathsf{a}_1 \rangle, \langle \mathsf{a}_2 \rangle\}$.

Again, let us notice that in our context, given a characterization for $\mathbf{U}$, call it φ, then we can easily construct a characterization for $\mathbf{U}^k$ of the form

$$x_1, ..., x_k \leftarrow \bigwedge_{s \in [2...k]} \top(x_s), \mathsf{twin}_s(x_1, x_s) \wedge \bigwedge_{\alpha \in atm(\varphi)} \alpha.$$

Vice versa, say that you have a characterization φ' for $\mathbf{U}^k$, then by renaming every occurrence of the free variables other than the first one, we will end up with a characterization for $\mathbf{U}$. Having asserted this, we can equivalently solve the problem by concentrating on the unary case.

Let $K_4 = (V_4, E_4)$ denote a representation of the complete graph of order four. Note that D_3 translates K_4 into a dataset, whereas $D_1 \ \cup \ D_4$ translates two disjoint objects isomorphic to G^a, call them G^{a_1} and G^{a_2}, into a dataset. Let $\tilde{a} \in \{a_1, a_2\}$. On the one hand, say that G is 3-colourable. From Lemma 1, $G^{\tilde{a}}$ is 4-colourable, so there exists a homomorphism h from $G^{\tilde{a}}$ to K_4. This homomorphism will map $\tilde{a}$ to a certain element of V_4, and since elements from V_4 are equal up to isomorphism, w.l.o.g. $h(\tilde{a}) = \mathsf{b}_1$. Now let φ be a characterization for $ess(\mathbf{U}, \mathcal{S})$. This means that there exists a homomorphism $h_1 : atm(\varphi) \mapsto \mathbb{D}(\mathcal{S})$ such that $h_1(x) = \tilde{\mathsf{a}}$. But then, by homomorphism composition, $h_2 : atm(\varphi) \mapsto \mathbb{D}(\mathcal{S})$ defined as $h \circ h_1$ is another homomorphism such that $h_2(x) = \mathsf{b}_1$, and so $\langle \mathsf{b}_1 \rangle \in ess(\mathbf{U}, \mathcal{S})$.

On the other hand, now say that $\langle \mathsf{b}_1 \rangle \in ess(\mathbf{U}, \mathcal{S})$. This means that $can(ess(\mathbf{U}, \mathcal{S}))$ explains V_4. In particular, this gives rise to a homomorphism from $G^{\tilde{a}}$ to K_4. But then, from Lemma 1, G is 3-colourable. As before,

Algorithm 4: $\mathsf{InEss}(\mathbf{U}, \mathcal{S}, \tau)$

Input: $\mathcal{S} = (K, \varsigma)$, $\mathbf{U} = \{\tau_1, ..., \tau_m\}$ and a tuple τ.

```
1 φ := BuildCan(U, S)
2 guess a mapping h from D_{atm(φ)} to D_{ς(τ)}
3 if h is a C-homomorphism from atm(φ) to ς(τ) then
4     if ⟨h(x_{s_1}), ..., h(x_{s_n})⟩ = τ then
5         accept this branch
6 reject this branch
```

what remains to be shown is that the new summary selector can be built without making complexity leaps and that its execution time turns out to be polynomial with respect to the input received. Again, this is straightforward because also in this case, the summary selector has always to return the entire dataset in input.

□

The decision problem ESS will in fact be the one we will use for every other reduction present in the rest of the work.

Gadgets

Before analyzing SIM, INC and PREC, we provide upper bounds for our gadgets. Both gadgets will be useful as we will see below to discuss the belonging of the next three decision-making problems that we will study in the relevant computational classes. Both problems reduce to ESS. Given an input $\iota = (\mathcal{S}, \mathbf{U}, \tau, \tau')$, to show GAD1 $\leq$ ESS we can use the reduction $\iota \mapsto (\mathcal{S}, \mathbf{U} \cup \{\tau'\}, \tau)$ and for GAD2 $\leq$ ESS we can use the reduction $\iota \mapsto (\mathcal{S}, \mathbf{U} \cup \{\tau\}, \tau')$. Let us see a formal proof below.

Proposition 15. *Both* GAD1 *and* GAD2 *belong to* NEXP, NP *and* P *in the broad, medium and handy case, respectively.*

Proof. Let $\iota = (\mathcal{S}, \mathbf{U}, \tau, \tau')$ be an input for GAD1. To show GAD1 $\leq$ ESS we can use the reduction $\iota \mapsto (\mathcal{S}, \mathbf{U} \cup \{\tau'\}, \tau)$, please note that this reduction preserves the hypothesys of the problem, in other word the broad case reduces to the broad case, the medium case and handy case reduce to general, medium case reduces to the medium case and handy case reduces to the handy case. The if and only if holds by definition. This, together with the upper bounds for ESS, in the broad, medium and handy case respectively, shown in Theorem 6 ends the first part of what we wanted to show. Let now

$\iota = (\mathcal{S}, \mathbf{U}, \tau, \tau')$ be an input for GAD2. To show GAD2 $\leq$ ESS we can simply use the reduction $\iota \mapsto (\mathcal{S}, \mathbf{U} \cup \{\tau\}, \tau')$ and then use a similar argument as before. □

We are not concerned with studying the possible completeness of these gadgets with respect to the computational classes they belong to since their only use in the following is aimed at giving upper bounds for the decision problems of our real interest. We can now complete our computational trip.

SIM

The idea behind the SIM decision problem is to be able to know whether two tuples, both different and not contained in the input unit, share the same nexus of similarity with the existing unit.

Theorem 7. SIM *is* NEXP*-complete,* NP*-complete and in* P *in the broad, medium and handy case, respectively. In particular, for each* $k > 0$*, LBs hold already for* SIM_k*.*

Proof. For what concerns the upper bounds, it is easy to notice that an input for SIM is accepted if, and only if, it is accepted both for GAD1 and GAD2. The following reduction is constructed in such a way as to preserve our assumptions. We will therefore give a single construction that will vary only for the nature of its input. As is natural, the broad case will be reduced starting from the broad case of the starting problem, while the medium case from the medium case of the starting problem.

Now let us face the lower bounds for this problem. Let $\text{ESS}_1^{\emptyset}$ be the decision problem ESS with the further hypothesis that the input unit $\mathbf{U}$ is unary, its cardinality is bigger than one, and the input ontology O is equal to the empty set. Then it holds that $\text{ESS}_1^{\emptyset} \leq \text{SIM}_k$ via the map $(\mathcal{S}, \mathbf{U}, \tau) \mapsto (\mathcal{S}', \mathbf{U}^k, \tau^k, \langle \mathsf{alias} \rangle^k)$, where:

- $\mathcal{S} = (K, \varsigma)$,
- $K = (D, O)$,
- $\mathbf{U} = \{\tau_1, ..., \tau_m\}$,
- $m > 1$,
- τ_1 is of the form $\langle c \rangle$,
- $\mathcal{S}' = (K', \varsigma')$,

- $K' = (D', O')$,
- $D' = \bar{D} \cup tw(\mathbb{D}_{\bar{D}}, k)$,
- $\bar{D} = double(D, c)$,
- $O = \emptyset$,
- $O' = \emptyset$,
- selector ς' is such that $\varsigma'(K', \tau') = \varsigma(K, \tau_0) \cup tw(\mathbb{D}_{\{\tau_0\}}, k)$,
- $\tau' \in \mathbb{D}^k_{D'}$
- $\tau_0 = \mathit{off}(\tau', c)$.

First, let us notice that in our context, given a characterization for $\mathbf{U}$, call it φ, then we can easily construct a characterization for $\mathbf{U}^k$ of the form

$$x_1, ..., x_k \leftarrow \bigwedge_{s \in [2...k]} \top(x_s), \mathsf{twin}_s(x_1, x_s) \wedge \bigwedge_{\alpha \in atm(\varphi)} \alpha.$$

Vice versa, say that you have a characterization for $\mathbf{U}^k$, then by renaming every occurrence of the free variables other than the first one, we will end up with a characterization for $\mathbf{U}$. Having asserted this, we can equivalently solve the problem by concentrating on the unary case.

The first thing the reader has to notice is that $\langle \mathsf{alias} \rangle \in ess(\mathbf{U}, \mathcal{S}')$ by construction. In fact, say that h is a $\mathcal{C}$-homomorphism from $atm(can(ess(\mathbf{U}, \mathcal{S})))$ to $\varsigma'(D', \langle c \rangle)$ such that $x_1 \mapsto c$. Please note that this $\mathcal{C}$-homomorphism must exist by hypotheses. Then we can construct another $\mathcal{C}$-homomorphism from $atm(can(ess(\mathbf{U}, \mathcal{S})))$ to $\varsigma'(D', \langle \mathsf{alias} \rangle)$, call it $\tilde{h}$, such that $x_1 \mapsto \mathsf{alias}$ as follows: $\tilde{h}(y) = h(y)$ if $y \neq x_1$ and $\tilde{h}(x_1) = \mathsf{alias}$. That being said, the problem collapses to whether or not $\tau \in ess(\mathbf{U}, \mathcal{S})$, and if that is the case, then we will end up with a yes answer since the same $\mathcal{C}$-homomorphism that was a witness for τ to be in $ess(\mathbf{U}, \mathcal{S})$ is still a witness for τ to be in $ess(\mathbf{U}, \mathcal{S}')$ and vice versa. Having a $\mathcal{C}$-homomorphism that is a witness for τ to be in $ess(\mathbf{U}, \mathcal{S}')$ leads us to a witness for τ to be in $ess(\mathbf{U}, \mathcal{S})$.

What remains to be shown is that the new summary selector can be built without making complexity leaps and that its execution time turns out to be polynomial with respect to the input received. This is evident, as we only have to do a linear scan of the input that allows us to reconstruct the initial dataset and build the new tuple on which to then relaunch the old summary selector. This second part will, by hypothesis, run in polynomial

time, and finally, we will add the other elements always doing a linear scan of the input.

□

An affirmative answer to this decision problem tells us that, from the point of view of the information contained in the unit, there is no way to distinguish the two tuples. In other words, if we ever encounter one of the two along our path of expansion surely, at the same moment, we will also encounter the other.

INC

The idea behind the SIM decision problem is to be able to know whether two tuples, both different and not contained in the input unit, share incomparable nexus of similarity with the existing unit.

Theorem 8. INC *is* coNEXP*-complete,* coNP*-complete and in* P *in the broad, medium and handy case, respectively. In particular, for each* $k > 0$*, LBs hold already for* INC_k*.*

Proof. For what concerns the upper bounds, it is easy to notice that an input for INC is accepted if, and only if, it is accepted both for ¬GAD1 and ¬GAD2. The following reduction is constructed in such a way as to preserve our assumptions. We will therefore give a single construction that will vary only for the nature of its input, as is natural, the broad case will be reduced starting from the broad case of the starting problem, while the medium case from the medium case of the starting problem.

For what concerns the lower bounds. Let $+\text{ESS}_1^{\emptyset}$ be the decision problem ESS with the further hypothesis that the input unit $\mathbf{U}$ is unary, its cardinality is bigger than one, and the input ontology O is equal to the empty set, then it holds that $+\text{ESS}_1^{\emptyset} \leq \neg\text{INC}_k$ via the following map: $(\mathcal{S}, \mathbf{U}, \tau) \mapsto (\mathcal{S}', \mathbf{U}^k, \tau^k, \langle \mathsf{alias} \rangle^k)$, where :

- $\mathcal{S} = (K, \varsigma)$,
- $K = (D, O)$,
- $O = \emptyset$,
- $\mathbf{U} = \{\tau_1, ..., \tau_m\}$,
- $m > 1$,
- τ_1 is of the form $\langle c \rangle$,

- $\mathcal{S}' = (K', \varsigma')$,
- $K' = (D', O')$,
- $D' = D \quad \cup \quad tw(\mathbb{D}_{\bar{D}}, k) \quad \cup \quad fc(\mathbb{D}_D)$,
- $\bar{D} = double(D, c)$,
- $O' = \emptyset$,
- ς' is such that $\varsigma'(D', \tau') = \varsigma(K, \tau_0) \quad \cup \quad tw(\mathbb{D}_{\{\tau_0\}}, c) \quad \cup \quad fc(\mathbb{D}_D)$,
- $\tau' \in \mathbb{D}^k_{D'}$
- $\tau_0 = off(\tau', c)$.

First, let us notice that in our context, given a characterization for $\mathbf{U}$, call it φ, then we can easily construct a characterization for $\mathbf{U}^k$ of the form

$$x_1, ..., x_k \leftarrow \bigwedge_{s \in [2...k]} \top(x_s), \mathsf{twin}_s(x_1, x_s) \wedge \bigwedge_{\alpha \in atm(\varphi)} \alpha.$$

Vice versa, say that you have a characterization for $\mathbf{U}^k$, then by renaming every occurrence of the free variables other than the first one, we will end up with a characterization for $\mathbf{U}$. Having asserted this, we can equivalently solve the problem by concentrating on the unary case.

By construction, we know that $\langle \mathsf{alias} \rangle$ for sure is not in $ess(\mathbf{U}, \mathcal{S}')$ and this is because the formula $x \leftarrow \mathsf{focus}(x)$ explains every element in $\mathbf{U}$ but does not explain $\langle \mathsf{alias} \rangle$. This lets us know that a characterization for $\mathbf{U}$ will use that predicate and so it will not explain $\langle \mathsf{alias} \rangle$. That being said, the problem collapses to whether or not $\tau \in ess(\mathbf{U}, \mathcal{S})$ or not and if that is the case then we will end up with a no answer since the same $\mathcal{C}$-homomorphism that was a witness for τ to be in $ess(\mathbf{U}, \mathcal{S})$ can be enlarged to a witness for τ to be in $ess(\mathbf{U}, \mathcal{S}')$ and vice versa. Having a $\mathcal{C}$-homomorphism that is a witness for τ to be in $ess(\mathbf{U}, \mathcal{S}')$ leads us to a witness for τ to be in $ess(\mathbf{U}, \mathcal{S})$.

What remains to be shown is that the new summary selector can be built without making complexity leaps and that its execution time turns out to be polynomial with respect to the input received. This is evident as we only have to do a linear scan of the input that allows us to reconstruct the initial dataset and build the new tuple on which to then relaunch the old summary selector, this second part will by hypothesis run in polynomial time, and finally, we will add the other elements always doing a linear scan of the input.

□

An affirmative answer to this decision problem tells us that, from the point of view of the information contained in the unit, there is no way to find the two tuples at the same time following the same expansion path. In other words, if we ever meet one of the two along our expansion path, the probability of meeting the other decreases, this is because having met one of the two we were only interested in the nexus of similarity that it shared with that tuple, leaving the others behind.

PREC

The idea behind the PREC decision problem is to be able to know whether two tuples, both different and not contained in the input unit, share different nexus of similarity with the given unit, with the further hypothesis that in particular the first tuple shares higher nexus of similarity with the unit.

Theorem 9. PREC *is* DEXP*-complete,* DP*-complete and in* P *in the broad, medium and handy case, respectively. In particular, for each* $k > 1$*, LBs hold already for* PREC_k*.*

Proof. Before throwing ourselves headlong into the proof, it is good to give the following definition and derive some important properties from it.

Let $\mathbf{x}_1$ and $\mathbf{x}_2$ be the sequences $x_1, \ldots, x_{n_1}$ and $x'_1, \ldots, x'_{n_2}$ respectively. A formula $\varphi \in \text{N}\mathbb{C}\text{F}$ is said *separable* if the following two conditions apply: the formula is of the form $\mathbf{x}_1, \mathbf{x}_2 \leftarrow \mathit{conj}(\varphi_1), \mathit{conj}(\varphi_2)$ where $\mathit{dom}(\mathit{atm}(\varphi_1)) \cap \mathit{dom}(\mathit{atm}(\varphi_2)) = \emptyset$ and both $\mathbf{x}_1 \leftarrow \mathit{conj}(\varphi_1)$ and $\mathbf{x}_2 \leftarrow \mathit{conj}(\varphi_2)$ are in $\text{N}\mathbb{C}\text{F}$.

Consider a separable formula φ of the form $\mathbf{x}_1, \mathbf{x}_2 \leftarrow \mathit{conj}(\varphi_1), \mathit{conj}(\varphi_2)$, a $|\mathbf{x}_1|$-ary tuple $\langle \mathbf{c}_1 \rangle$, a $|\mathbf{x}_2|$-ary tuple $\langle \mathbf{c}_2 \rangle$, and a selective knowledge base $\mathcal{S}$. If $\varsigma(\langle \mathbf{c}_1, \mathbf{c}_2 \rangle) = \varsigma(\langle \mathbf{c}_1 \rangle) \cup \varsigma(\langle \mathbf{c}_2 \rangle)$, and $\varsigma(\langle \mathbf{c}_1 \rangle) \cap \varsigma(\langle \mathbf{c}_2 \rangle) = \emptyset$, then it holds that $\langle \mathbf{c}_1, \mathbf{c}_2 \rangle \in \mathit{inst}(\varphi, \mathcal{S})$ if, and only if, $\langle \mathbf{c}_1 \rangle \in \mathit{inst}(\varphi_1, \mathcal{S})$ and $\langle \mathbf{c}_2 \rangle \in \mathit{inst}(\varphi_2, \mathcal{S})$.

The previous observation simply derives from the fact that by construction, and thanks to the hypothesis $\varsigma(\langle \mathbf{c}_1, \mathbf{c}_2 \rangle) = \varsigma(\langle \mathbf{c}_1 \rangle) \cup \varsigma(\langle \mathbf{c}_2 \rangle)$ we are able to work separately on the two pieces. In particular, we are able to materialize two distinct C-homomorphisms, call them h_1 and h_2, one from $\mathit{atm}(\varphi_1)$ to $\varsigma(\langle \mathbf{c}_1 \rangle)$ and the other from $\mathit{atm}(\varphi_2)$ to $\varsigma(\langle \mathbf{c}_2 \rangle)$ whose union gives rise to a third C-homomorphism from $\mathit{atm}(\varphi)$ to $\varsigma(\langle \mathbf{c}_1, \mathbf{c}_2 \rangle)$.

Call this next statement (i), we will use it at the end of the proof.

Consider now a separable formula φ of the following form $\mathbf{x}_1, \mathbf{x}_2 \leftarrow \mathit{conj}(\varphi_1), \mathit{conj}(\varphi_2)$, a $(|\mathbf{x}_1| + |\mathbf{x}_2|)$-ary $\mathbf{U}$, and a selective knowledge base $\mathcal{S}$. Let $\mathbf{U}_1$ be the $|\mathbf{x}_1|$-ary unit of the form $\{\tau \ : \ \exists \tau' \ s.t \ \tau \circ \tau' \in \mathbf{U}\}$ and $\mathbf{U}_2$ be the $|\mathbf{x}_2|$-ary unit of the form $\{\tau \ : \ \exists \tau' \ s.t \ \tau' \circ \tau \in \mathbf{U}\}$. If $\forall \tau \in \mathbf{U}$ we have $\varsigma(\tau) = \varsigma(\tau') \cup \varsigma(\tau'')$ and $\varsigma(\tau') \cap \varsigma(\tau'') = \emptyset$, where $\tau = \tau' \circ \tau''$ with $\tau' \in \mathbf{U}_1$

and $\tau'' \in \mathbf{U}_2$, then it holds that φ is a characterization for $\mathbf{U}$ if, and only if, φ_1 is a characterization for $\mathbf{U}_1$ and φ_2 is a characterization for $\mathbf{U}_2$.

For what concerns the upper bounds, it is easy to notice that an input for PREC is accepted if, and only if, it is accepted both for GAD1 and $\neg$GAD2. The following reduction is constructed in such a way as to preserve our assumptions. We will therefore give a single construction that will vary only for the nature of its input; as is natural, the broad case will be reduced starting from the broad case of the starting problem, while the medium case from the medium case of the starting problem.

Let us now talk about lower bounds.

Let $+\text{ESS}_1^{\emptyset}$ be the decision problem ESS with the further hypothesis that the input unit $\mathbf{U}$ is unary, its cardinality is bigger than one, and the input ontology O is equal to the empty set. Let $+\text{ESS}_{k-1}^{\emptyset}$ be the decision problem ESS with the further hypothesis that the input unit $\mathbf{U}$ is $(k-1)$-ary, its cardinality is bigger than one, and the input ontology O is equal to the empty set.

For each $L \in \mathsf{DEXP}$ (resp., DP), $L \leq \text{PREC}_k$ in the broad (resp., medium) case, via the mapping $w \mapsto (\mathcal{S}, \mathbf{U}, \tau', \tau'')$, where w is any string over the alphabet of L, $L = L_1 \cap L_2$, $L_1 \in \mathsf{NEXP}$ (resp., NP), $L_2 \in \mathsf{coNEXP}$ (resp., coNP), ρ_1 is a reduction from L_1 to $+\text{ESS}_1^{\emptyset}$ in the broad (resp., medium) case, ρ_2 is a reduction from L_2 to $\neg+\text{ESS}_{k-1}^{\emptyset}$ in the broad (resp., medium) case, each $\rho_i(w) = (\mathcal{S}_i, \mathbf{U}_i, \tau_i)$, each $\mathcal{S}_i = (K_i, \varsigma_i)$, each $K_i = (D_i, \emptyset)$, without loss of generality each constant and predicate different from $\top$ in D_1 (resp., D_2) has a- (resp., b-) as a prefix, each $\mathbf{U}_i = \{\tau_{i,1}, ..., \tau_{i,m_i}\}$, $\mathcal{S} = (K, \varsigma)$, $K = (D, \emptyset)$, $D = D_1 \cup D_2$, ς is such that $\varsigma(K, \tau) = \varsigma_1(K_1, \tau|_{\mathbb{D}(D_1)}) \cup \varsigma_2(K_2, \tau|_{\mathbb{D}(D_2)})$, $\mathbf{U} = \mathbf{U}_1 \times \mathbf{U}_2$, $\tau' = \tau_1 \circ \tau_{2,1}$, $\tau'' = \tau_{1,1} \circ \tau_2$, each $\tau|_{\mathbb{D}(D_i)}$ is obtained from τ by removing the terms not in $\mathbb{D}(D_i)$.

The main idea of the construction is that the final knowledge base that we construct is actually bipartite in two distinct senses. It is not bipartite just from the domain point of view, but in particular, the predicate $\top$ is the only one shared by both types of constants. Apart from that, constants coming from the first reduction are not just totally different from constants occurring in the second reduction, but they also appear in atoms with totally different predicate names.

Noting this fact gives us the possibility to say that the explanation we will construct will be necessarily separable in the sense that we defined above.

From this, everything will follow from point (i). What remains to be shown is that the new summary selector can be built without making com-

plexity leaps and that its execution time turns out to be polynomial with respect to the input received. But this is just a combination of the running of two distinct procedures that we do know, by hypotheses, run in polynomial time. Moreover, we are able to easily reconstruct both D_1 and D_2 via a linear scan of the input.

□

An affirmative answer to this decision problem tells us that somehow the first tuple "must come before" the other in every possible expansion that we are going to follow, in other words there is no way to encounter the second tuple in any path of expansion if we have not found the other first and/or at the same time.

Computational Aspects Summary

This chapter explored a comprehensive range of computational problems within our formal framework, focusing on the nexus of similarity among tuples of entities and the tractability of these problems under different theoretical assumptions. Fundamental definitions and notations were introduced, serving as the foundation for understanding the computational tasks and their connections to the framework.

The chapter discussed the procedural construction of canonical and core characterizations ($can(\mathbf{U}, \mathcal{S})$ and $core(\mathbf{U}, \mathcal{S})$), analyzing their computational complexity and the impact of various constraints on their size and the number of possible homomorphisms. Specific algorithms for computing these characterizations were detailed, along with their steps and computational costs.

Furthermore, the chapter defined decision tasks such as PREC, SIM, and INC, which compared the nexus of similarity between tuples, and GAD1 and GAD2, which served as analytical tools. The tractability of these tasks was examined under three computational assumptions: broad, medium, and handy, each imposing different limitations on the arity of predicates, the size of the initial unit, and the domain of summaries.

The chapter concluded with a focus on essential notations and reductions, providing intuitive explanations and examples, and drawing upon established literature to support the complexity analyses. It emphasized the practical significance and applications of the decision tasks defined, ensuring clarity and a strong theoretical foundation for the computational aspects discussed.

Chapter 7

A Prototypical Tool

In the preceding chapter, we introduced key reasoning tasks that stand at the forefront of our investigative endeavor. As we progress, this chapter is dedicated to unveiling a prototypical tool, crafted in the Java programming language, which serves as an instrumental asset in identifying nexus characterizations, both canonical and core.

Moreover, the tool addresses the key reasoning tasks, providing a concrete solution that enhances our comprehension of the nexus characterization.

Starting from this tool, we have successfully derived and analyzed a series of statistics that clarify the correlation between the sizes of the canonical and core characterizations in relation to the changing cardinality of the input unit. This investigation effectively confirms the practical need to employ core characterizations, elevating them from theoretical novelty to a vital tool for improving human understanding of the results.

7.1 A First Implementation

The reasoning tasks discussed in Chapter 6 have all been implemented in a prototypical tool.

The tool, which can be found at https://tinyurl.com/mt9zaeeu, is to be considered as a first implementation capable of solving the computational tasks we defined previously in a typical setting. It is therefore far from being in its final form and should only be understood as a proof of concept of the effectiveness of the approach, as well as a useful tool for validating what we consider to be the fundamental building blocks of our framework.

Purpose and Limitations

Currently, the tool is designed to address the tasks outlined in Chapter 6, operating on the premise that the unit being analyzed for unveiling the nexus of similarity beetwen its tuples and the additional tuples being compared for similarity levels—whether higher, equivalent, or incomparable—have all arity one. Additionally, it is configured to process datasets where the predicates have an arity of two.

What we give as input to the tool is a unary unit $\mathbf{U}$, a dataset D with predicate arity equal to exactly two, an ontology O (it can also be empty if we do not want to derive new knowledge starting from the input dataset), a summary selector ς compatible with both the dataset and the ontology, and (optionally) two additional unary tuples τ_1 and τ_2 of which we want to know if the nexus of similarity they share with the input unit relative to the selective knowledge base that we are in fact considering are higher, coincide, or are incomparable with each other. In Section 7.2, there will be a comprehensive example to better guide the reader.

Input Description

As we have already mentioned previously in the section, we need to give as input to the tool a unary unit $\mathbf{U}$, a dataset D with predicate arity equal to exactly two, an ontology O, a summary selector ς compatible with both the dataset and the ontology, and (optionally) two additional unary tuples τ_1 and τ_2. Below we show how the specific input can be presented to the tool and in order to be clearer we will present the same example shown in Chapter 1, in particular in Section 1.2, obviously adapted to the needs of the case. Lets begin by expressing how to give the input dataset. One of the most natural ways to express the initial input dataset D, without considering the knowledge that we will derive from the ontology, is to write everything in a list of datalog facts. Datalog is a well-known logic programming language that allows us to represent facts and rules about a domain of interest. Fundamentally, a list of datalog facts is nothing else than a simple form of extensional knowledge that describes the data in the database.

Example 34. *What we need to do in order to express the input dataset is writing a list of datalog facts in the following way:*

isA(“Epcot”, “theme park”).
isA(“theme park”, “amusement park”).
isA(“Discovery Cove”, “theme park”).
isA(“Pacific Park”, “amusement park”).
isA(“Gardaland”, “theme park”).
isA(“Leolandia”, “amusement park”).
isA(“Prater”, “amusement park”).
located(“Prater”, “Austria”).
located(“Pacific Park”, “California”).
located(“Discovery Cove”, “Florida”).
located(“Epcot”, “Florida”).
located(“Leolandia”, “Italy”).
located(“Gardaland”, “Italy”).
partOf(“Florida”, “US”).
partOf(“California”, “US”).

■

Secondly, we can express the input ontology O as a short list of datalog rules. As we know, datalog rules are statements that define new relations from existing relations using logical inference. Fundamentally, a list of datalog rules can be seen as a simple form of intensional knowledge that describes the ontology in the database.

Example 35. *If we take a look at our Example 1, what we need to do in order to express the input ontology is nothing more than writing the only rule we need in order to transitively close the* isA *relation, in other terms the following rule:*

$$\mathsf{isA}(\mathsf{X},\ \mathsf{Y}) :- \mathsf{isA}(\mathsf{X},\ \mathsf{Z}),\ \mathsf{isA}(\mathsf{Z},\ \mathsf{Y}).$$

■

Finally, we need a way to describe the summary selector ς. A natural way is to use datalog rules again and with the next example we will show one of the possible translations. Since, in principle, summary selectors are functions that take as input, in addition to the reference tuple τ, a set of facts and return a subset of facts that are relevant or interesting for the user (and related to τ), using datalog rules that specify conditions or criteria for selecting the facts is probably the most natural way in which a summary selector can be defined.

Example 36. *Since in the example of the initial chapter we are interested in taking all the facts that have at most "distance two" from our reference entity (distance calculated as we were moving in a directed graph and facts containing our entity in first position are considered to distance zero, facts containing in the first position the entities in the second position of the facts at distance 0 are considered at distance one and so on). One of the simplest things to do is to first translate everything into a single predicate, for convenience let us call it* triple *of arity three, in which we will actually translate all the facts obtained, starting from the action of the ontology on the initial dataset in union with the initial dataset itself.*

$$\begin{aligned}&\mathsf{triple}(\mathsf{X},\ \text{“isA”},\ \mathsf{Y}) : -\mathsf{isA}(\mathsf{X},\ \mathsf{Y}).\\&\mathsf{triple}(\mathsf{X},\ \text{“partOf”},\ \mathsf{Y}) : -\mathsf{partOf}(\mathsf{X},\ \mathsf{Y}).\\&\mathsf{triple}(\mathsf{X},\ \text{“located”},\ \mathsf{Y}) : -\mathsf{located}(\mathsf{X},\ \mathsf{Y}).\end{aligned}$$

At this point, we can indicate to our logical program which objects should be treated as entities. In general, we are interested in summaries of tuples and not of entities but this set of rules is sufficient to cover the initial example we built in Chapter 1. One possible way of expressing how to identify an entity is given by the following two rules:

$$\begin{aligned}&\mathsf{entity}(\mathsf{X}) : -\mathsf{triple}(\mathsf{X},\ \_,\ \_).\\&\mathsf{entity}(\mathsf{Y}) : -\mathsf{triple}(\_,\ \_,\ \mathsf{Y}).\end{aligned}$$

Now we need a way to tell when and if a fact is part of the summary of a given unary tuple or not. A possible way to imitate the summary defined in Chapter 1 is given by the following set of 4 rules:

$$\begin{aligned}&\mathsf{inSummary\_unary}(\mathsf{X},\ \mathsf{X},\ \mathsf{P},\ \mathsf{Y}) :- \mathsf{triple}(\mathsf{X},\ \mathsf{P},\ \mathsf{Y}).\\&\mathsf{inSummary\_unary}(\mathsf{X},\ \mathsf{X},\ \text{“entityOf”},\ \text{“top”}) :- \mathsf{entity}(\mathsf{X}).\\&\mathsf{inSummary\_unary}(\mathsf{X},\ \mathsf{Y},\ \text{“entityOf”},\ \text{“top”}) :- \mathsf{inSummary\_unary}(\mathsf{X},\ \_,\ \_,\ \mathsf{Y}),\\&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathsf{Y} <> \text{“top”}.\\&\mathsf{inSummary\_unary}(\mathsf{X},\ \mathsf{Y},\ \mathsf{R},\ \mathsf{Z}) :- \mathsf{inSummary\_unary}(\mathsf{X},\ \mathsf{X},\ \mathsf{P},\ \mathsf{Y}),\\&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathsf{triple}(\mathsf{Y},\ \mathsf{R},\ \mathsf{Z}),\ \mathsf{P} <> \text{“isA”},\\&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathsf{R} <> \text{“isA”}.\end{aligned}$$

It is now a good idea to explain what the four previous rules do and why they are sufficient to cover the cases of interest to us. The first rule is in fact collecting all the facts at a distance of 0 from our entity, i.e. all the atoms in which it holds the first position which semantically speaking indicates its role as "subject" with respect to the action. The second rule is imposing that

in each summary of each entity X *there is a fact, of arity two, translatable with* entityOf(X, "top")*, in fact this object it will represent* $\top$(X)*, it is just a different way of representing it and it is convenient for us because we can thus maintain few rules and not deal with predicates of distinct arity on a case-by-case basis. Obviously this is a possible translation, but nothing prevents us from treating the predicate* $\top$ *separately with ad hoc rules. In any case, given the non-ambiguity due to the knowledge of the domain and the non-existence of other atoms of the type* entityOf(X, Y) *this was one of the simplest and most immediate ways to pursue. The third rule ensures that we take a representation of all atoms of the type* $\top$(Y) *for every entity that is obviously different from* "top"*, we do this because the entity* "top" *is a fictitious entity artificially introduced by us for convenience but it was not part of the initial dataset nor could it be derived starting from the ontology at our disposal. The attentive reader will have noticed that with this third rule we are only considering elements that are in the second position of the atoms, i.e. elements that semantically speaking behave as an "object" and not as a subject. Why is this enough? Well, because as we have defined our starting summary so that an entity* Y*, different from the entity* X *present in the unary tuple for which we want to explicitly calculate the summary, appears as subject in at least one of the facts that we must return and which is at a distance* n *from the entity* X*, then we know with certainty that* Y *has covered the role of object in at least one done at a distance* $n-1$ *from our reference entity. Last but not least, the fourth rule ensures that we are going to get all the facts at a distance of one, or, as we said in the initial Chapter, all the facts obtainable by following a path of at most two arcs starting from our reference entity. Finally, both the reference unit* $\mathbf{U}$ *and the two entities belonging respectively to the tuple* τ_1 *and* τ_2 *will be inserted into specific files but without the use of datalog rules. This is because the reasoning starting from this information, such as the construction of* $can(\mathbf{U}, \mathcal{S})$*, is not implemented starting from logical rules.* ■

Output Description

Depending on whether the user decides (i) to not give the two additional tuples as input or (ii) to do so, the tool will return some files containing some important notions. If the user chooses option (i), not to give the two additional tuples as input, they will receive a single output file. This file will contain the description of the unit received with all the individual unit tuples inside, the summaries of all the tuples relating to the selective knowledge base data in input, both $can(\mathbf{U}, \mathcal{S})$ and $core(\mathbf{U}, \mathcal{S})$ (although without the atoms of the form $\top(y)$ for readability issues), and last, but

not least $ess(\mathbf{U}, \mathcal{S})$.

On the other hand, if the user chooses (ii), together with the first file, there will be three more. If you define $\mathbf{U}' = \mathbf{U} \cup \{\tau_1\}$ and $\mathbf{U}'' = \mathbf{U} \cup \{\tau_2\}$, the user will also receive two files. One for each of the units, containing information relative to $\mathbf{U}'$ and $\mathbf{U}''$ respectively, in the same fashion as the first file. Moreover, there will also be another file containing the description of whether the nexus of similarity that one of the two tuples shares with $\mathbf{U}$ is higher, coincides, or is incomparable to each other.

Two important points of discussion emerge:

a) How, starting from logical rules, can we calculate which one of the two tuples τ_1 and τ_2 shares higher nexus of similarity with the initial unit or if they coincide or are incomparable with each other?

b) How to calculate $core(\mathbf{U}, \mathcal{S})$ always starting from logical rules.

How to resolve decision problems Regarding point *a*), the approach used was the following: first suppose that there exist $\mathsf{p_1(X)}$ and $\mathsf{p_2(X)}$ which respectively collect $inst(can(\mathbf{U} \cup \tau_1, \mathcal{S})) = inst(can(\mathbf{U}', \mathcal{S}))$ and $inst(can(\mathbf{U} \cup \tau_2, \mathcal{S})) = inst(can(\mathbf{U}'', \mathcal{S}))$. These two unary predicates are calculated at runtime following the procedure described in Chapter 4.

```
inc :− p1(X), not p2(X), p2(Y), not p1(Y).
notPrecOne :− p1(X), not p2(X).
notPrecTwo :− p2(Y), not p1(Y).
precOne :− not inc, notPrecTwo.
precTwo :− not inc, notPrecOne.
sim :− not inc, not precOne, not precTwo.
```

Now, let us delve into the details to understand how and why these six simple rules suffice to calculate the relationships between the nexus of similarity that the two tuples τ_1 and τ_2 share with the initial unit.

The role of the first rule is evident: the connections can only be incomparable with each other if, by definition, there exists at least one element captured by $\mathsf{p_1}$ and not by $\mathsf{p_2}$ and vice versa.

The second and third rules serve as counterparts to each other. They ensure that the first tuple cannot share a higher nexus of similarity with the unit than the second can, and vice versa. How can we assert this? It is straightforward. To guarantee that τ_1 does not share higher nexus of similarity with the unit than τ_2, it suffices to find an element captured by $\mathsf{p_1}$ but not by $\mathsf{p_2}$. Similarly, to ensure that τ_2 does not share higher nexus of similarity with the unit than τ_1, we find an element captured by $\mathsf{p_2}$ but not by $\mathsf{p_1}$.

The fourth and fifth rules also act as counterparts but with a different focus. They ensure that the first tuple shares higher nexus of similarity with the unit than the second and vice versa. To establish that τ_1 shares higher nexus of similarity with the unit than τ_2, we need to verify that the nexus of similarity are not incomparable, a guarantee provided by the negation of the first rule. Additionally, we confirm that τ_2 cannot share higher nexus of similarity with the unit than τ_1, a condition ensured by the possible activation of the third rule.

Conversely, to assert that τ_2 shares higher nexus of similarity with the unit than τ_1, we verify that the nexus of similarity are not incomparable, ensured by the negation of the first rule. Furthermore, we confirm that τ_1 cannot share higher nexus of similarity with the unit than τ_2, a condition guaranteed by the potential activation of the second rule.

If none of the first, fourth, and fifth rules are activated, the only remaining possibility is that the nexus of similarity shared by τ_1 with the unit coincide with those shared by τ_2. This situation leads to the formulation of the sixth rule.

How to calculate $core(\mathbf{U}, \mathcal{S})$ At run time a single logical program is written to be able to understand which are and which are not the fundamental atoms to maintain starting from the $can(\mathbf{U}, \mathcal{S})$ which is constructed procedurally following the construction shown in Section 4.3. In order to maintain high readability, we will show and explain what the program does starting from any formula, in which for simplicity the arity of the predicates is exactly equal to two, future implementations of the program will provide the ability to deal with arbitrary arity. The following example will help us to show all this.

Example 37. *Let φ be the formula $x \leftarrow \mathsf{r}(x, y_1), \mathsf{r}(x, y_2), \mathsf{r}(x, c)$ where $c \in \mathsf{C}$. Obviously, up to isomorphism, the core of φ is equal to $x \leftarrow \mathsf{r}(x, c)$. the program we will write will look like the following set of datalog facts and rules. To begin, we give the facts below and explain their meaning, then we will give the rules in detail.*

$$\begin{aligned}&\mathsf{rel}(\mathsf{u},\ \mathsf{r},\ \mathsf{y}_1,\ 1).\\ &\mathsf{rel}(\mathsf{u},\ \mathsf{r},\ \mathsf{y}_2,\ 2).\\ &\mathsf{rel}(\mathsf{u},\ \mathsf{r},\ \mathsf{c},\ 3).\end{aligned}$$

This first part simply translates the three atoms contained in $atm(\varphi)$ into datalog facts. The reader will have noticed that the predicate has arity 4 and that the last entry is in ascending order, this is not a random fact as those elements will serve us as a sort of "id" for the atoms, which will be

clear later. Let us now consider and comment on the following two rules and then, once the functioning of these two is clear, we can easily comment on the rest of the logical program.

$$\mathsf{rel1(X,\ Y,\ Z)} :- \mathsf{rel(X,\ Y,\ Z,\ \_)},\ \mathsf{not\ rel(X,\ Y,\ Z,\ 1)}.$$
$$\mathsf{rem1(X,\ Y,\ Z)} :- \mathsf{rel(X,\ Y,\ Z,\ 1)},\ \mathsf{rel1(u,\ r,\ Y_1)}, \mathsf{rel1(u,\ r,\ Y_2)}, \mathsf{rel1(u,\ r,\ c)}.$$

What we have done with the first rule is nothing more than creating an auxiliary predicate, called rel1*, which collects all the atoms of the initial formula with the exception of the first. In the following we will see how this procedure will be generalized to the $i-th$ atom. The second rule, however, does something different. In particular, it allows us to understand whether or not the atom with "id" equal to one is a "fundamental" atom. In other words, an atom that must necessarily belong to the core of the formula or not. This check is done in the following way. In fact, the predicate* rel1 *coming from the previous rule collects indifferently objects that should be treated as variables and objects that should be treated as constants all in the same way, and obviously treats them all like constants. In our rewriting, we are treating as constants only our free variable, which we called u to remind us that that element is part of the starting unit* $\mathbf{U}$*, and all the objects that must "really" be treated as such, i.e. the names of predicates that appear in the second position of the facts and the constants that appeared in the formula, all the other objects that were treated as constants, but in reality had to be considered as existentially quantified variables (e.g.* $\mathsf{y_1}$ *and* $\mathsf{y_2}$*) are rewritten in the rule so that the logic program recognizes them as variables. Why do we do this? It is easy to say. In fact, the reader will have noticed that in the predicate* rel1 *we have not collected all the facts present in* rel *and yet in the second rule we are acting as "if we had done so". So what does the possible activation of this second rule indicate? If the second rule is activated, given that the fact* $\mathsf{rel(X,\ Y,\ Z,\ 1)}$ *always exists and therefore cannot be the cause of a possible non-activation, it means that there exists a homomorphism from the entire formula, i.e. from the atoms represented by the facts of the predicate* rel*, to the formula without the first atom, i.e. to the atoms represented by the facts of the predicate* rel1*, and when this happens, in fact, it means that the first atom was not really necessary. In other words, it is not an atom that we need in the core and therefore we can mark it as a "removable" atom, hence it will be collected in the predicate* rem1*. Moreover, it is the only possible atom that can be collected by this predicate. This is in fact guaranteed by the id present at the beginning of the rule. Now we must ask ourselves how it is possible to generalize the same procedure to the $i-th$ atom. Below we will show the 4 rules that almost*

entirely compose the rest of the logical program and from this it will be clear what the correct way to do it is. Immediately afterwards we will comment on why they are really suitable for carrying out the intended task.

```
rel2(X, Y, Z) :- rel(X, Y, Z, _), not rel(X, Y, Z, 2), not rem1(X, Y, Z).
rem2(X, Y, Z) :- rel(X, Y, Z, 2), rel2(u, r, Y1), rel2(u, r, Y2),
                 rel2(u, r, c).
rel3(X, Y, Z) :- rel(X, Y, Z, _), not rel(X, Y, Z, 3), not rem1(X, Y, Z),
                 not rem2(X, Y, Z).
rem3(X, Y, Z) :- rel(X, Y, Z, 3), rel3(u, r, Y1), rel3(u, r, Y2),
                 rel3(u, r, c).
```

As can be seen, both the predicate rel2 *and the predicate* rel3 *eliminate not only the atom with id equal to two and the atom with id equal to 3 respectively, but also, when they were suitable for removal, all the atoms that we have already decided to eliminate. In this way, when we check for the existence of homomorphism, we are sure that this is compatible with the choices already made previously. If we did not in this way, we could risk eliminating many more atoms than necessary. Last but not least, we will present the rule that deals with collecting the atoms that will actually compose the core that we want to return, immediately after we will comment on the rule.*

```
core(X, Y, Z) :- rel(X, Y, Z, _), not rem1(X, Y, Z), not rem2(X, Y, Z),
                 not rem3(X, Y, Z).
```

In practice, with this last rule we are doing nothing other than maintaining all the initial atoms that have never been marked as "removable" and therefore are really necessary and are part of our core. ■

7.2 Benchmark Presentation

In light of the absence of established golden standards in the field, it becomes imperative to emphasize the necessity of introducing a benchmark for the nexus of similarity. It is important to clarify that the primary objective of this benchmark is not to serve as a definitive golden standard, but rather to offer a platform for the development and testing of various proof-of-concept methodologies. Given the inherent complexities and subjective nature of the nexus of similarity, it would be premature and perhaps unrealistic to establish a singular gold standard at this stage.

What we indeed anticipated from the utilization of this benchmark was the swift and conspicuous delineation of disparities, both in terms of magnitude and comprehensibility, between a canonical characterization and its core. This, in turn, effectively validates the significance of the core as something more than a mere stylistic exercise, but rather as an indispensable cornerstone of our approach. The transparent demonstration of these differences underscores the core's fundamental role in our framework, establishing it as an essential building block that is integral to the very essence of our approach.

All of the aforementioned points will, however, be further elucidated in the subsequent Section 7.3 following the discussion of results concerning an initial experimental assessment. This discussion will serve to solidify the connections between our conceptual framework and the real-world applications, offering a richer context for the points raised earlier in this thesis.

The dataset contains information about 130 films that are among the highest-grossing or most awarded films of the last 30 years. For each film, the dataset includes the following information:

- The title
- The year of release
- The director
- The screenwriter
- The production company
- The main actor
- The country where it was launched
- The genre
- The rating

The dataset also contains information about people who are among the most influential or recognized actors, directors, screenwriters, or producers in the film industry, and were all involved in those 130 films. For each person, the dataset includes the following information

- The name
- The nationality

- The role (actor, director, screenwriter, producer, ecc)

All the information in this dataset has been reported as factual statements that relate two entities by a specific predicate. The predicate is the name of the relation that holds between the two entities, such as directedBy, writtenBy, starring, citizenOf, and so on. The entities are the names of the films, people, companies, countries, genres, or other categories that are involved in the film domain. The predicate and the two entities form a binary relation that has arity 2, meaning that it has two arguments. For example, the fact that Steven Spielberg is the director of Indiana Jones and the Temple of Doom is represented by the binary relation directedBy("Indiana Jones and the Temple of Doom","Steven Spielberg"), which has arity 2. The dataset consists of a collection of such binary relations that describe various aspects of the film industry.

7.3 Experimental Analysis

What this experimental analysis aims to achieve is to point out the substantial differences in terms of the number of atoms and consequent readability between a canonical and a core type characterization. Simultaneously, it aims to highlight how core characterizations are still simple to obtain in terms of time when the summaries are kept reasonably small. These experiments are not the main focus of the discussion but should be understood as support for the theoretical analysis and as a bridge between our choices in the framework and a possible final implementation that is not only machine-readable but also human-readable.

As previously stated, the goal of this study is to estimate the parameter expressing the ratio between the number of atoms present in a canonical characterization and those present in a core characterization of a unit. To achieve this, eight random samples were used, each corresponding to units with a different cardinality of the reference unit. The samples cover units with reference units of cardinality two, three, and four, as well as an aggregate variable considering all units regardless of the cardinality of the reference unit. The last four samples are subsets extracted from the first ones, with the additional hypothesis that the essential expansion of the reference unit had lower cardinality than the entire dataset. These samples were drawn from populations of units with the respective cardinalities.

All data used can be found at https://tinyurl.com/mt9zaeeu.

In statistics, one of the objectives is to draw conclusions about a population based on a sample. However, the sample may not accurately represent

the population, and estimates may vary depending on the sample size and the sampling method. To account for this uncertainty, the *central limit theorem* [48] and *confidence intervals* [61, 62] can be employed.

The central limit theorem states that sufficiently large samples from a population will have sample means that follow a normal distribution, regardless of the shape of the population distribution. This allows the use of the normal distribution to calculate the probability of obtaining a certain sample mean given the population mean and standard deviation. On the other hand, a confidence interval is an interval containing the population parameter with a specified level of confidence. The confidence level indicates the proportion of times an interval includes the true value if the sampling process is repeated and is typically set at 95% or 99%.

By utilizing the central limit theorem and confidence intervals, statistical inferences about a population parameter, in this case, the ratio between a canonical and a core characterization, can be made based on a sample statistic. This helps in quantifying sampling variability and uncertainty.

We denote X_1 as the random sample representing the parameter related to the ratio between canonical and core characterizations when a unit has cardinality two, X_2 for cardinality three, X_3 for cardinality four, and X_4 as an aggregated random sample of all units. Additionally, four variants, denoted as X_5, X_6, X_7, and X_8, were derived from the previous samples with the additional condition that the cardinality of the essential expansion of the reference unit was lower than the entire dataset. These variants aim to observe how the pattern highlighted in the data changes as the number of nexus of similarity shared by the tuples in the reference unit varies.

Using unbiased estimators to calculate the mean and variance, if θ_i represents the parameter relating to the ratio between the number of atoms present in a canonical characterization compared to a core one in the sampling X_i, the results are presented in Table 7.1.

We also wanted to conduct another series of experiments in which we focus on formulas similar to characterizations but without the atoms of the type $\top(y)$, where y can be either a variable or a constant indifferently. This experiment was carried out in order to evaluate the possible impact for a more user-friendly visualization, since, in practice, the existence of the predicate $\top$ is a way to ensure without any doubt the existence of the characterizations, but it does not provide additional information on the other nexus of similarity. In some way, "every object knows that it belongs to the top family".

With this idea in mind, the samplings $\tilde{X_1}$,...,$\tilde{X_8}$ were born which are nothing other than the same samplings X_1,...,X_8 from which however, the

Parameter	Confidence Interval 95%	Confidence Interval 99%
θ_1	$[5.35, 5.46]$	$[5.33, 5.48]$
θ_2	$[20.02, 20.69]$	$[19.92, 20.79]$
θ_3	$[70.40, 73.65]$	$[69.89, 74.16]$
θ_4	$[31.95, 33.24]$	$[31.76, 33.43]$
θ_5	$[4.33, 4.55]$	$[4.29, 4.58]$
θ_6	$[25.06, 27.84]$	$[24.63, 28.27]$
θ_7	$[139.14, 167.53]$	$[134.74, 171.93]$
θ_8	$[20.98, 24.55]$	$[20.43, 25.10]$

Table 7.1: Parameters for the random samplings X_1,..., X_8 and their respective confidence intervals at 95% and 99%.

atoms belonging to the $\top$ predicate have been eliminated.

Also in this case using the appropriate unbiased estimators to calculate the mean and variance, if we call $\tilde{\theta_i}$ the parameter relating to the ratio between the number of atoms present in a canonical characterization compared to a core one in the sampling $\tilde{X_i}$, we obtain the results in table 7.2. We should however not be misled by a naive interpretation of the data at

Parameter	Confidence Interval 95%	Confidence Interval 99%
$\tilde{\theta_1}$	$[7.16, 7.33]$	$[7.14, 7.35]$
$\tilde{\theta_2}$	$[28.67, 29.63]$	$[28.53, 29.78]$
$\tilde{\theta_3}$	$[103.31, 108.02]$	$[102.72, 108.62]$
$\tilde{\theta_4}$	$[46.42, 48.29]$	$[46.18, 48.53]$
$\tilde{\theta_5}$	$[4.81, 5.07]$	$[4.78, 5.09]$
$\tilde{\theta_6}$	$[29.13, 32.43]$	$[28.83, 32.73]$
$\tilde{\theta_7}$	$[165.13, 198.74]$	$[161.89, 201.98]$
$\tilde{\theta_8}$	$[24.53, 28.76]$	$[24.21, 29.09]$

Table 7.2: Parameters for the random samplings $\tilde{X_1}$,..., $\tilde{X_8}$ and their respective confidence intervals at 95% and 99%.

our disposal. The non-normality of the initial distribution, which we can notice from the following plots representing the various distributions just discussed, makes it that referring to the mean of the parameter studied is, although correct, misleading from the point of view of intuition.

The following description applies to each of the following graphs.

The bar chart shows the frequency of samples with different values of the atom ratio, defined as the quotient between the number of atoms in a canonical characterization and the number of atoms in a core character-

ization. The x-axis is divided into 20 equal bins, ranging from 0 to the maximum possible value of the atom ratio. The y-axis shows the number of samples that have an atom ratio within each bin. The bar chart illustrates how the atom ratio varies across different samples and characterizations.

The attentive reader will have noticed that in most samplings we find ourselves faced with the phenomenon known as *Skewed Data* [38]. We would have various ways to rebalance the work. We could in fact adapt the sampling through a transformation capable to bring the data back into normal form. This would allow us to give an intuition closer to reality on the true meaning of the average of the parameters studied previously. However the parameter itself would lose its meaning. For this last reason, one of the ways that best suits our needs is to use statistical tests with simpler direct interpretation. For example, the percentiles, which in our case reveal the results reported in table 7.3.

What can we say by looking at this data and why was it not enough to exploit the central limit theorem? The fundamental point is that in situations such as those before us the average, however accurate, is not an intuitively valid representative for the population. This is because skewed data, by their nature, tends not to agglomerate around their average, and this is what causes our intuition on the matter to falter. So why is exploiting percentiles instead a more appropriate choice?

Even if they are a less sophisticated tool, in practice they give us a clear vision of the minimum possible variation that we should expect, and from them it is clearer than ever that even in the rosiest of expectations dictated by the random sample X_5. However more than 50% of the population has a canonical characterization that is more than 3 times larger than its corresponding core, and on the other hand we know that random populations exist, such as for example the X_3 sample in which almost 99% of the population sees a ratio of almost 6 to 1 between the size of the canonical characterization and that of its core type counterpart. Furthermore, for the same sample we can see how 50% of the population shows a ratio higher than 43 to 1 between the size of the canonical characterization and that of its core counterpart, not to mention the fact that there are samples like X_7 and $\tilde{X}_7$ in which 50% of the population exhibits a ratio even higher than 128 to 1.

In the final analysis of this chapter, it is paramount to highlight the profound importance of the subsequent studies [1, 22] with respect to practical applications. These works are not merely academic exercises but are pivotal for envisioning and actualizing real-time applications. They utilize BabelNet [60] as an essential semantic resource, which will be further enhanced

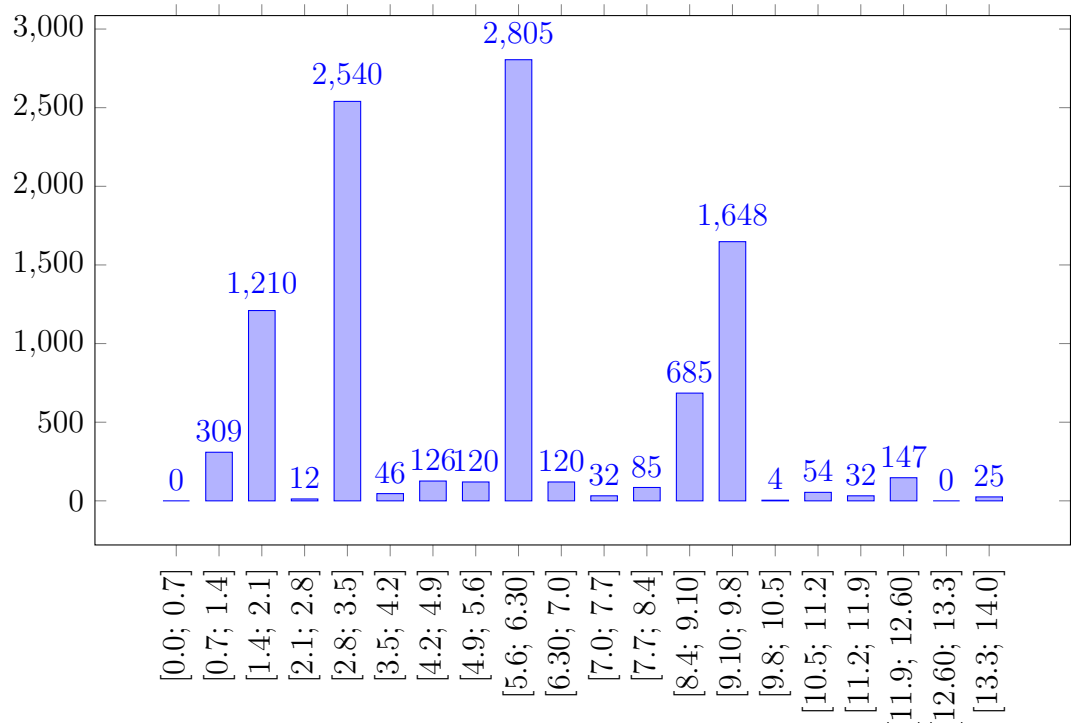

Figure 7.1: Distribution of the ratio between the number of atoms in a canonical characterization and its core-type counterpart in random sampling X_1. This random sample is, together with the sample $\tilde{X}_1$ for obvious reasons, the only one with respect to which the study of the mean value as a reference does not appear to be misleading from the point of view of intuition. As the reader will have noticed, this is due to the fact that the data seem to be distributed with a certain regularity below the classic bell which represents the normal distribution. It is also evident that a large part of the population has positioned itself extremely close to the mean value itself.

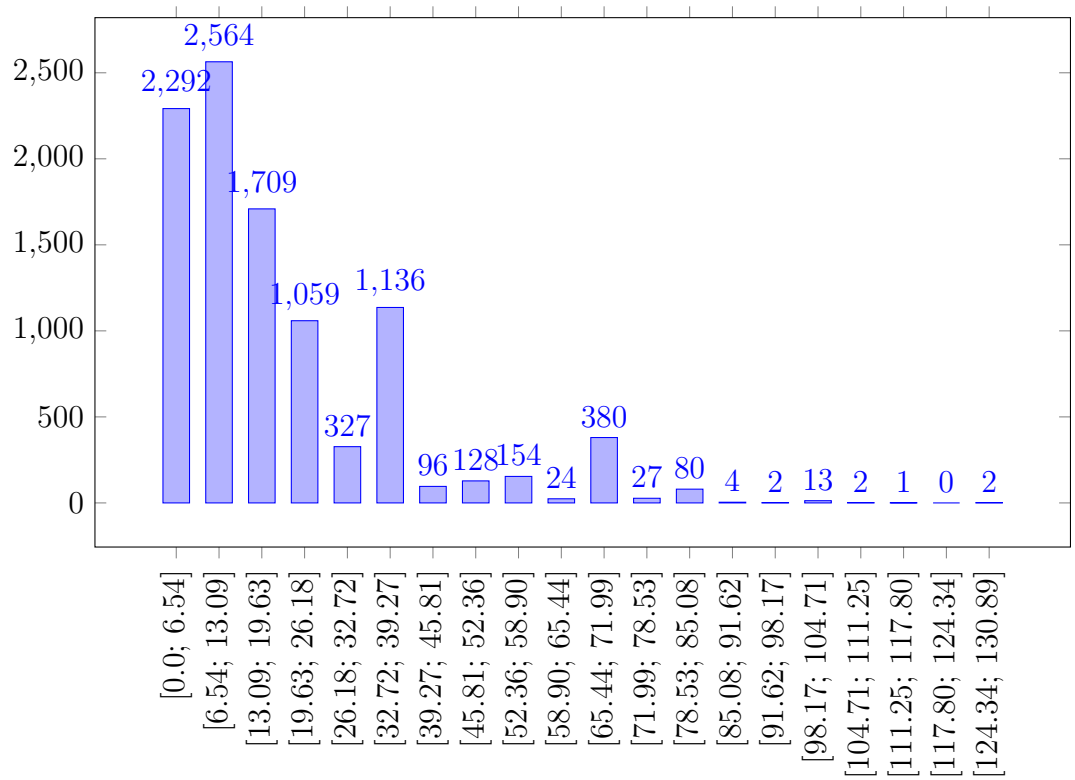

Figure 7.2: Distribution of the ratio between the number of atoms in a canonical characterization and its core-type counterpart in random sampling X_2.

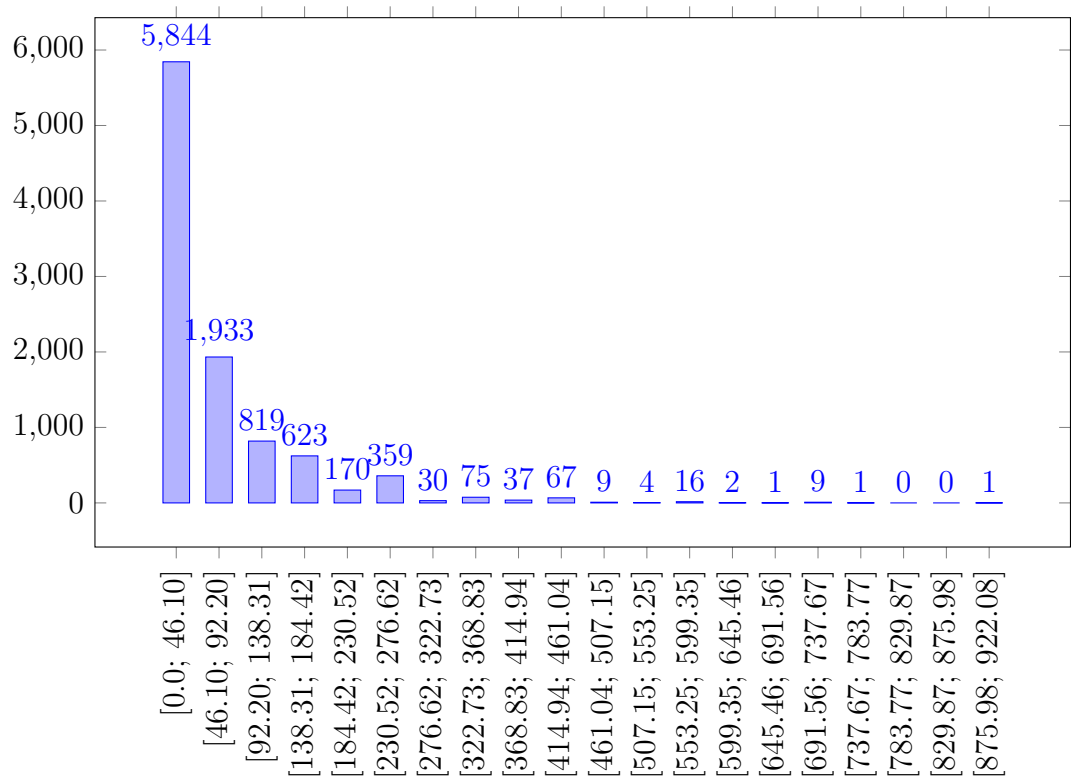

Figure 7.3: Distribution of the ratio between the number of atoms in a canonical characterization and its core-type counterpart in random sampling X_3.

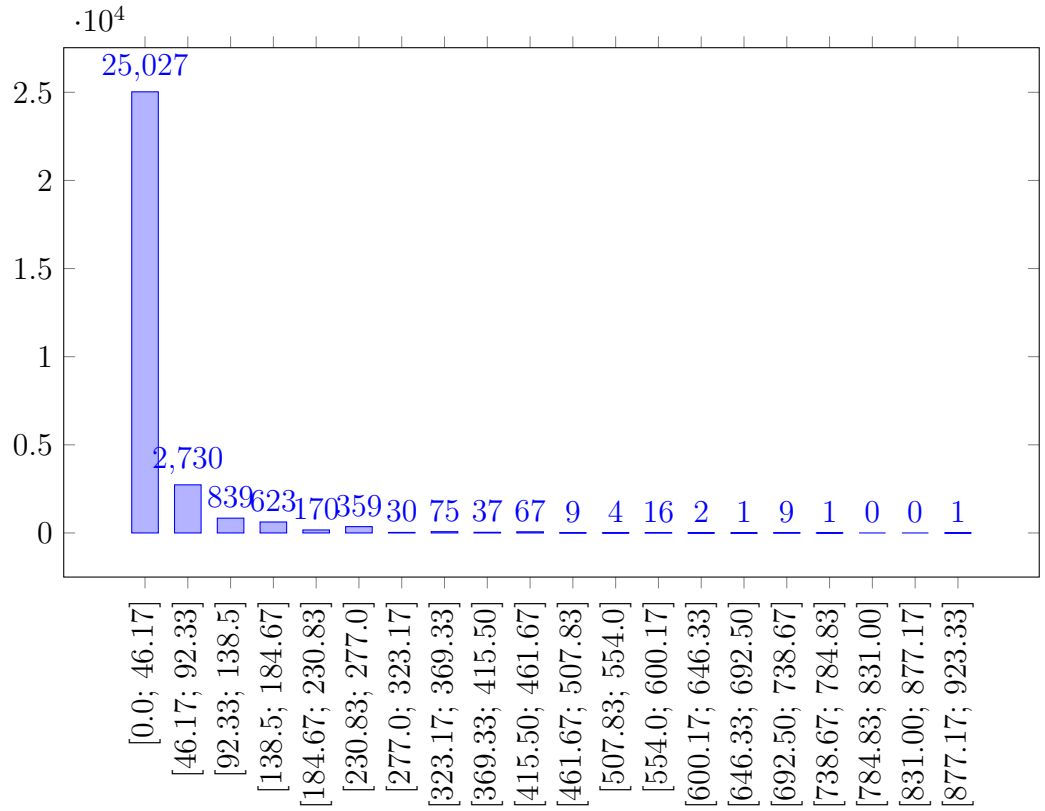

Figure 7.4: Distribution of the ratio between the number of atoms in a canonical characterization and its core-type counterpart in random sampling X_4.

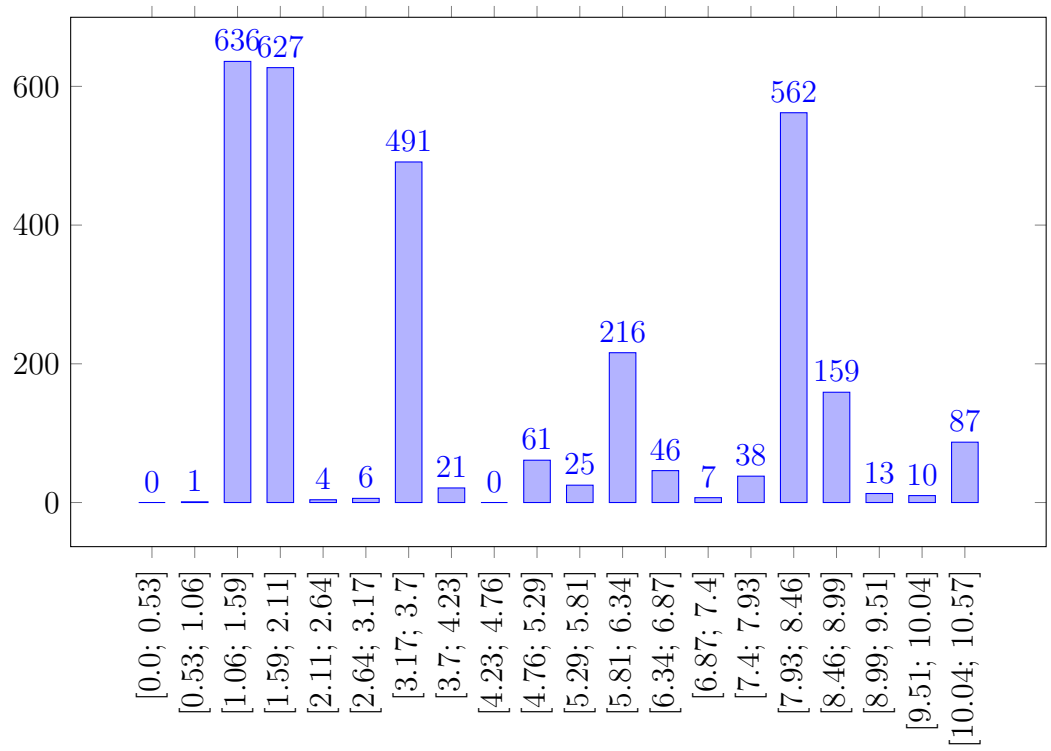

Figure 7.5: Distribution of the ratio between the number of atoms in a canonical characterization and its core-type counterpart in random sampling X_5.

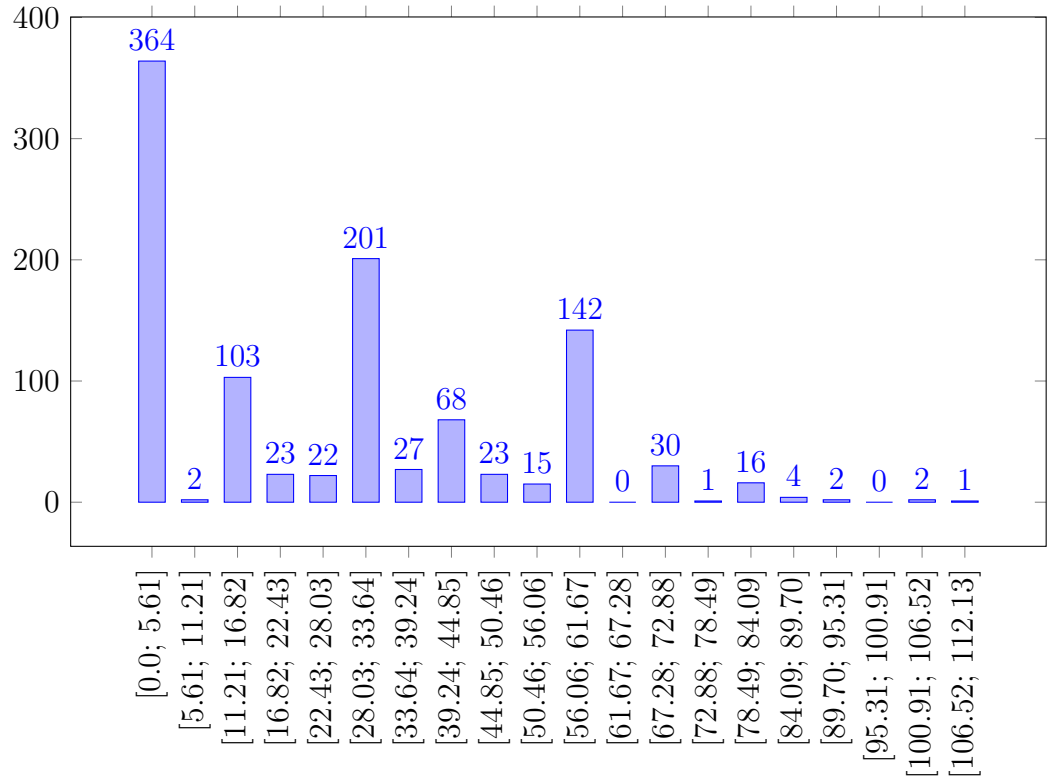

Figure 7.6: Distribution of the ratio between the number of atoms in a canonical characterization and its core-type counterpart in random sampling X_6.

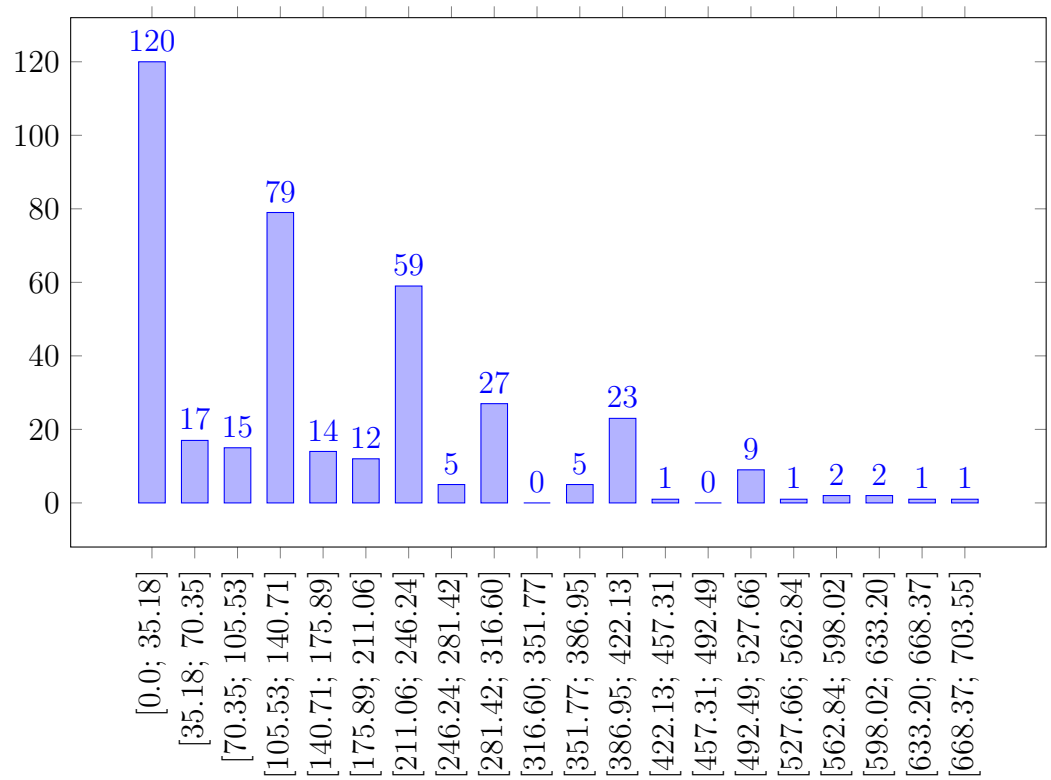

Figure 7.7: Distribution of the ratio between the number of atoms in a canonical characterization and its core-type counterpart in random sampling X_7.

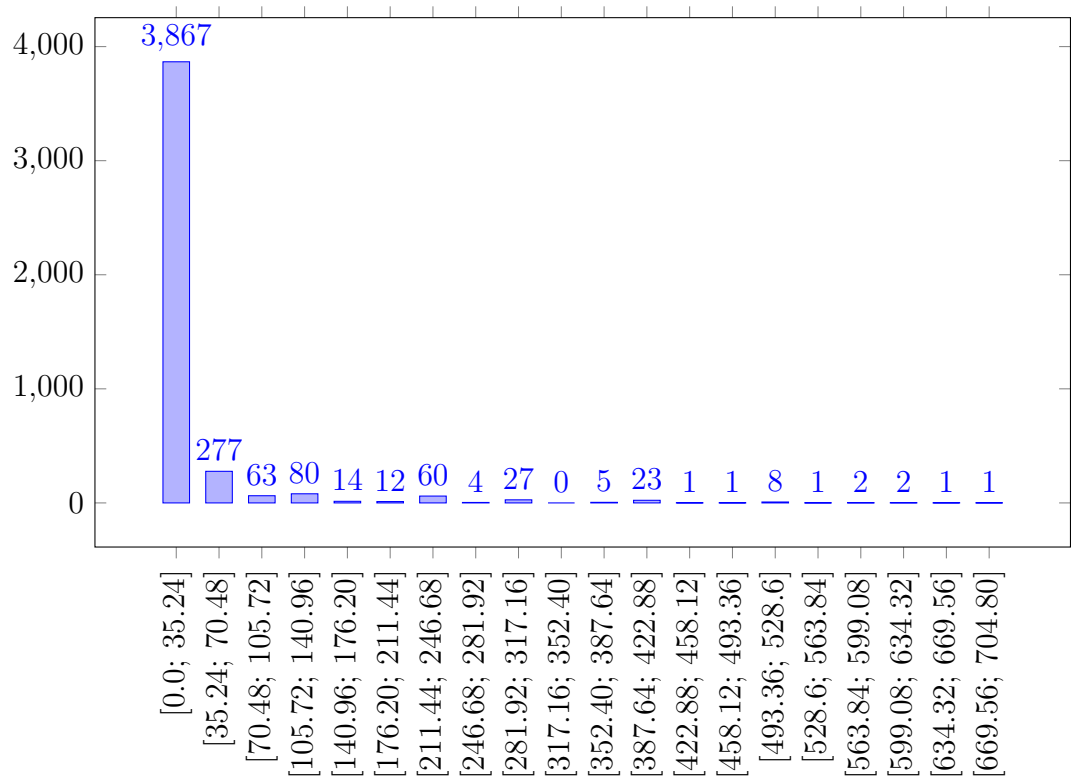

Figure 7.8: Distribution of the ratio between the number of atoms in a canonical characterization and its core-type counterpart in random sampling X_8.

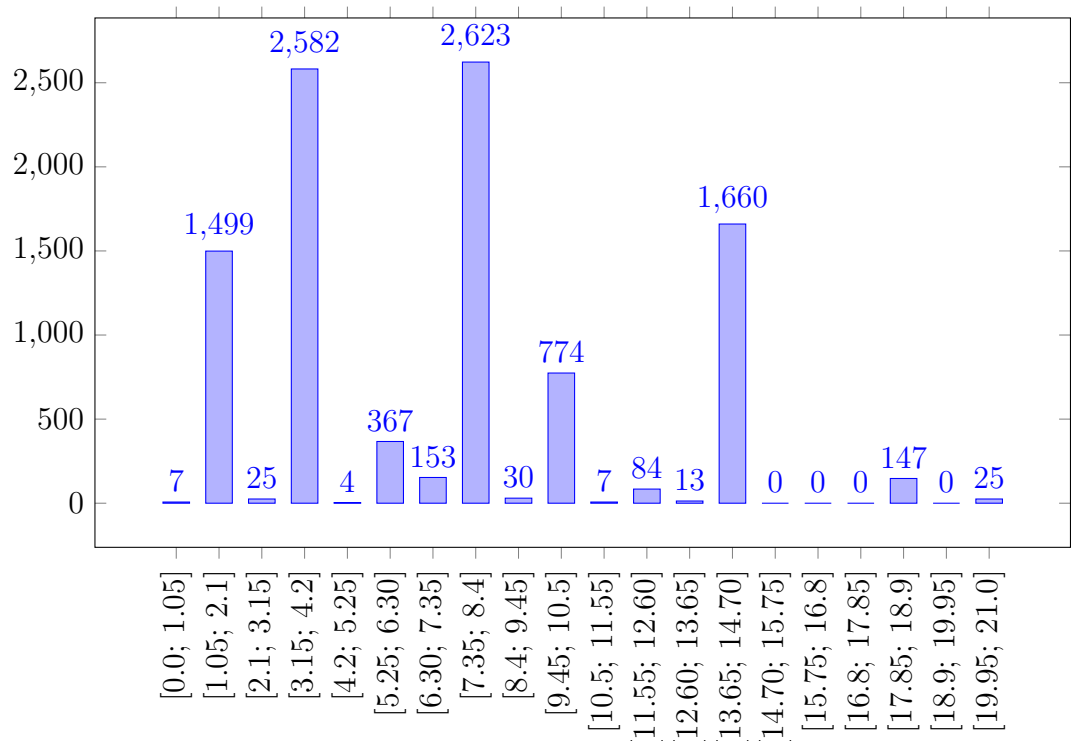

Figure 7.9: Distribution of the ratio between the number of atoms in a canonical characterization and its core-type counterpart in random sampling $\tilde{X}_1$. This random sample is, together with the sample X_1 the only one with respect to which the study of the mean value as a reference does not appear to be misleading from the point of view of intuition. As the reader will have noticed, this is due to the fact that the data seem to be distributed with a certain regularity below the classic bell which represents the normal distribution. It is also evident that a large part of the population has positioned itself extremely close to the mean value itself.

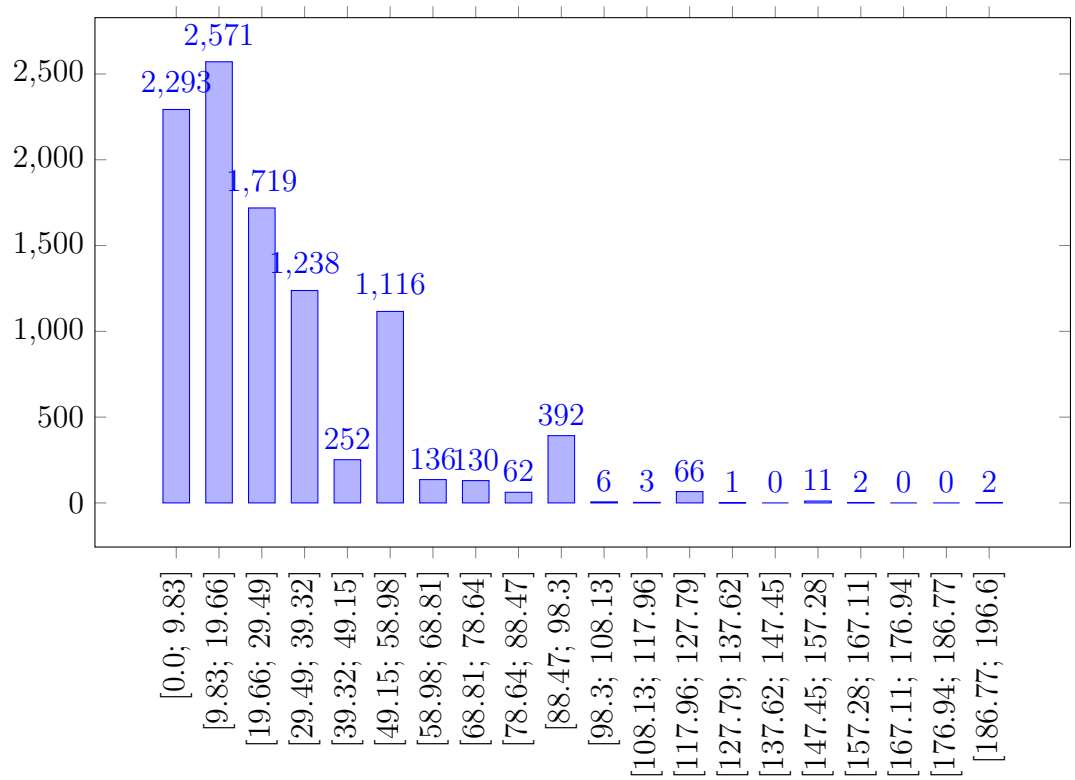

Figure 7.10: Distribution of the ratio between the number of atoms in a canonical characterization and its core-type counterpart in random sampling $\tilde{X}_2$.

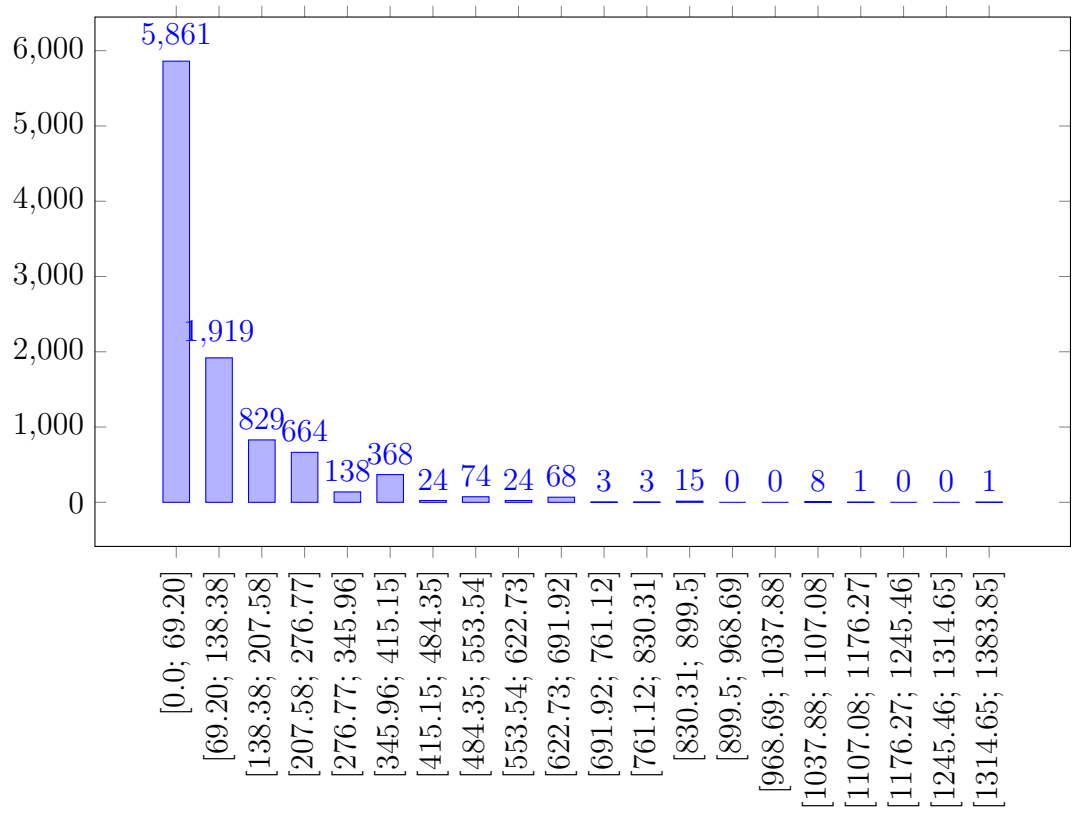

Figure 7.11: Distribution of the ratio between the number of atoms in a canonical characterization and its core-type counterpart in random sampling $\tilde{X}_3$.

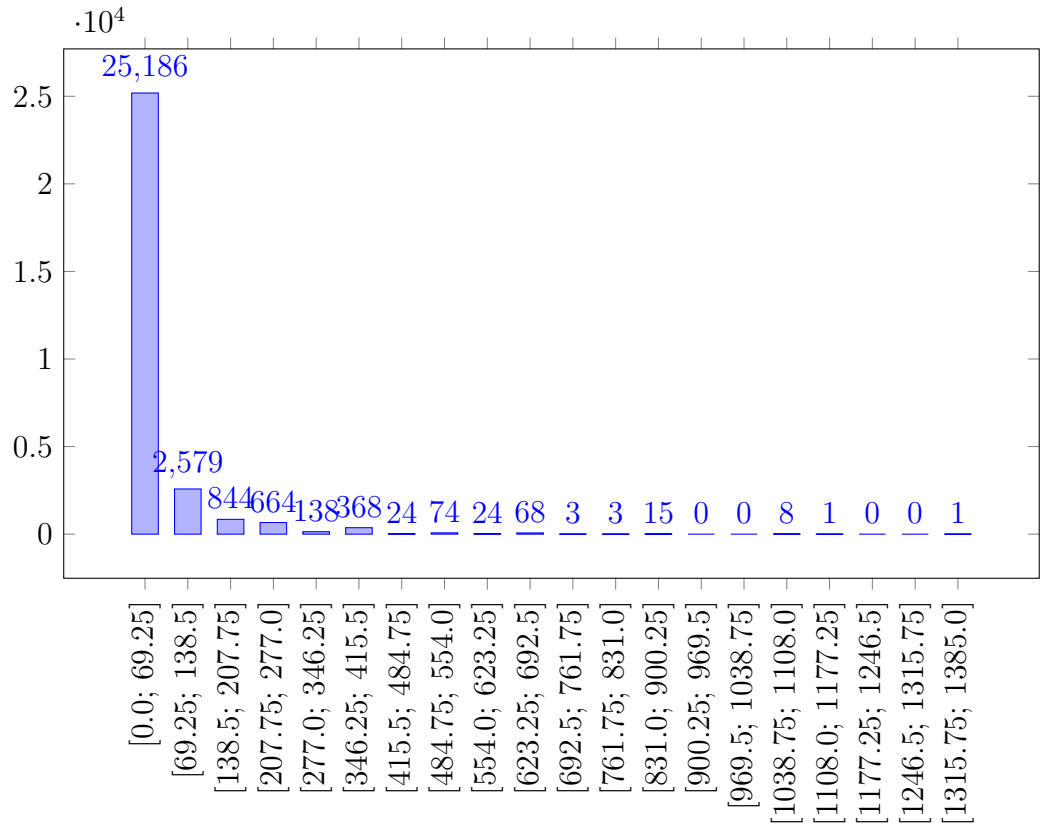

Figure 7.12: Distribution of the ratio between the number of atoms in a canonical characterization and its core-type counterpart in random sampling $\tilde{X}_4$.

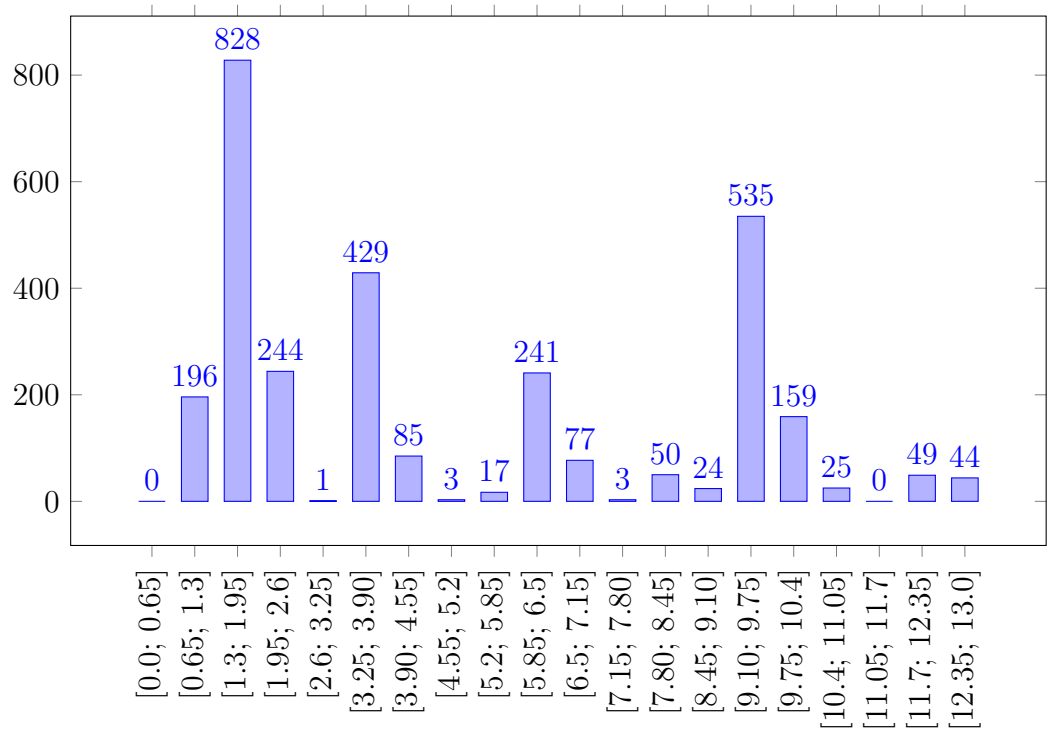

Figure 7.13: Distribution of the ratio between the number of atoms in a canonical characterization and its core-type counterpart in random sampling $\tilde{X}_5$.

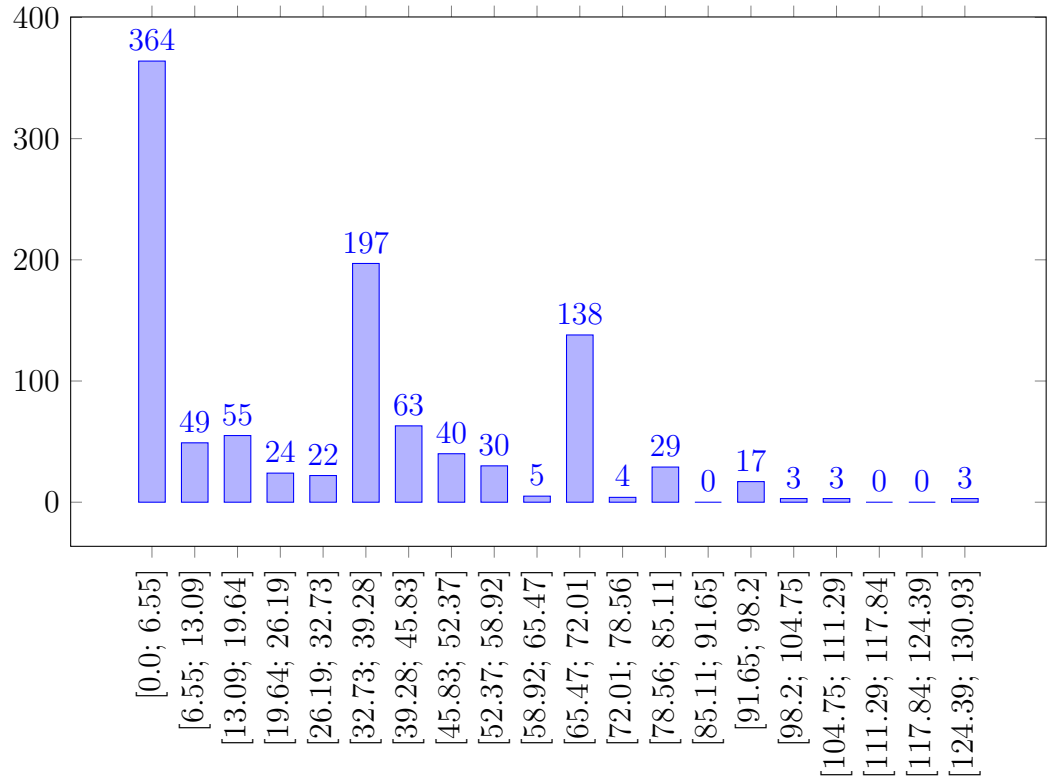

Figure 7.14: Distribution of the ratio between the number of atoms in a canonical characterization and its core-type counterpart in random sampling $\tilde{X}_6$.

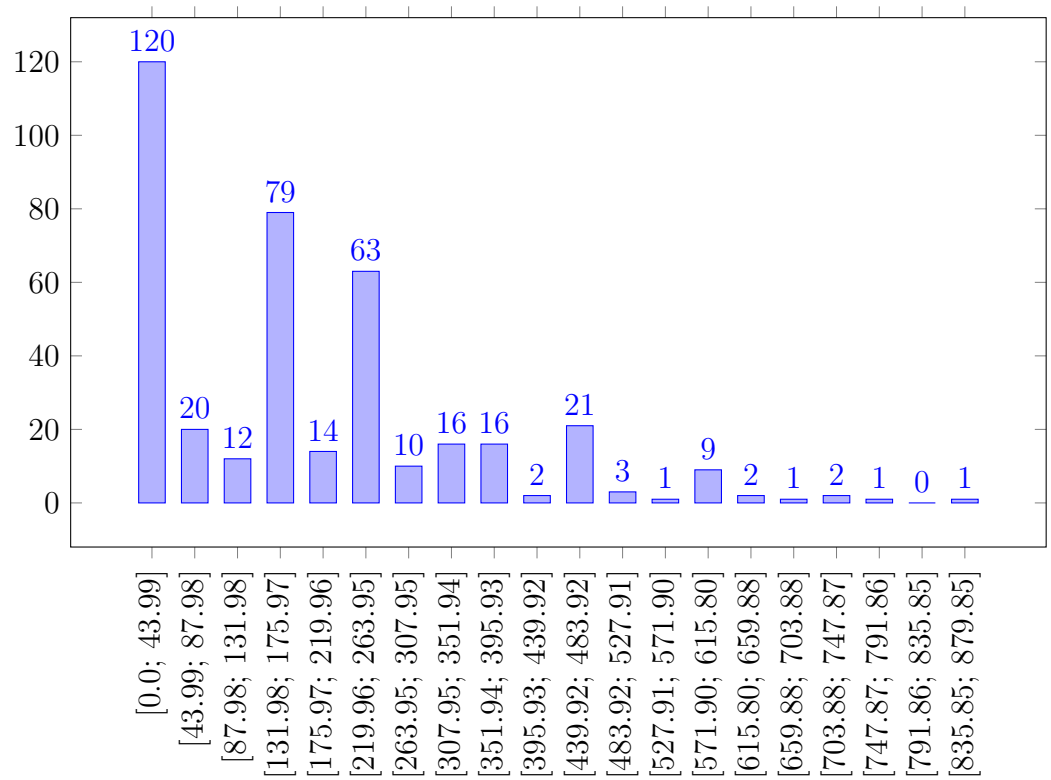

Figure 7.15: Distribution of the ratio between the number of atoms in a canonical characterization and its core-type counterpart in random sampling $\tilde{X}_7$.

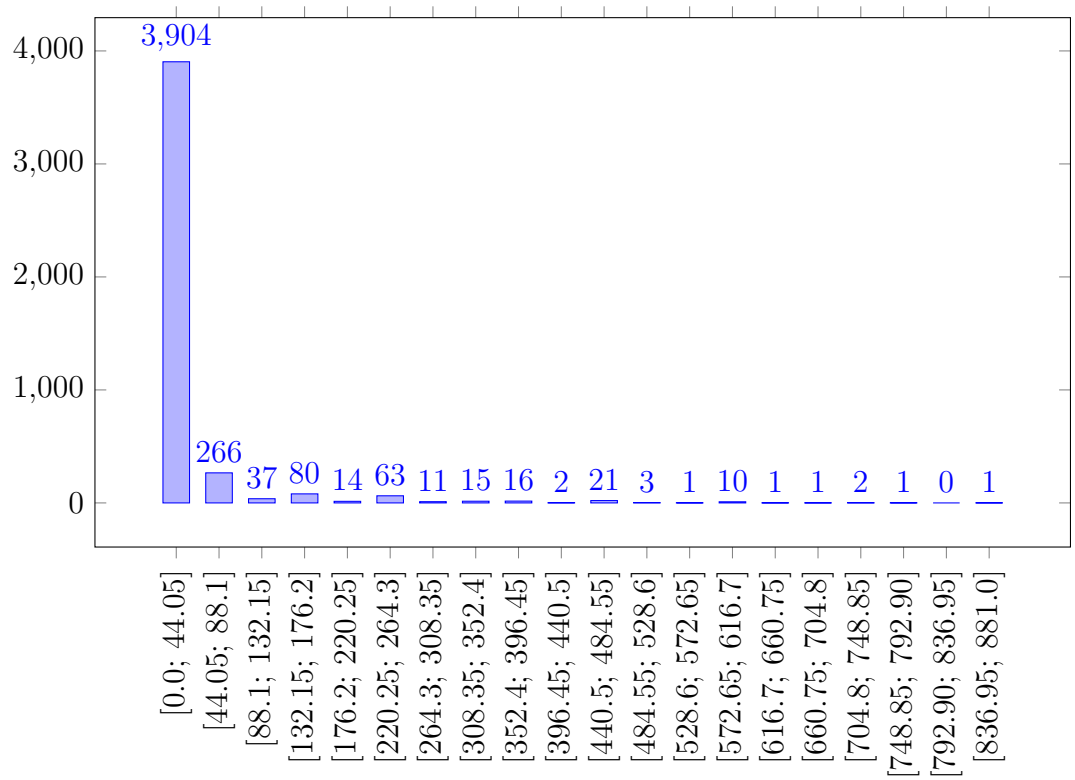

Figure 7.16: Distribution of the ratio between the number of atoms in a canonical characterization and its core-type counterpart in random sampling $\tilde{X}_8$.

Sampling	1-st	5-th	10-th	25-th	50-th
X_1	$1,23$	$1,50$	$1,67$	$3,00$	$5,67$
X_2	$2,17$	$3,20$	$5,67$	$11,00$	$16,33$
X_3	$5,67$	$11,00$	$11,00$	$21,67$	$43,00$
X_4	$1,40$	$1,83$	$3,00$	$5,67$	$11,00$
X_5	$1,17$	$1,25$	$1,40$	$1,67$	$3,18$
X_6	$1,72$	$1,92$	$2,20$	$3,13$	$32,43$
X_7	$2,75$	$3,60$	$4,12$	$7,30$	$128,71$
X_8	$1,17$	$1,35$	$1,48$	$1,80$	$5,00$
$\tilde{X}_1$	$1,23$	$1,46$	$1,85$	$4,00$	$8,00$
$\tilde{X}_2$	$2,15$	$4,00$	$8,00$	$14,00$	$24,00$
$\tilde{X}_3$	$7,62$	$16,00$	$16,00$	$32,00$	$64,00$
$\tilde{X}_4$	$1,38$	$2,00$	$4,00$	$8,00$	$16,00$
$\tilde{X}_5$	$1,15$	$1,25$	$1,38$	$1,62$	$3,40$
$\tilde{X}_6$	$1,69$	$1,92$	$2,21$	$3,31$	$37,67$
$\tilde{X}_7$	$2,75$	$3,58$	$4,23$	$8,08$	$150,00$
$\tilde{X}_8$	$1,15$	$1,33$	$1,46$	$2,00$	$5,67$

Table 7.3: The table above shows the corresponding random samples and the relative percentiles.

by the forthcoming integration of Wikidata [83]. The provision of concise summaries will only augment their value.

These studies serve as the cornerstone for the ambitious implementations that we are committed to developing in the subsequent year.

Chapter 8

Related Work

This chapter will explore the different approaches to artificial intelligence, focusing on two main categories: symbolic and subsymbolic. Symbolic approaches are based on the use of logic, rules, and symbols to represent and manipulate knowledge. Subsymbolic approaches are based on the use of numerical values, vectors, matrices, and functions to model and learn from data. The chapter will be divided into two main sections, namely "Topics with Symbolic Approaches" and "Topics with Subsymbolic Approaches". The former will cover topics related to first-order logic and knowledge representation. The latter will cover topics such as Semantic similarity, entity set expansion and knowledge graph embeddings.

8.1 Topics with Symbolic Approaches

In this section, we will examine some of the symbolic approaches to artificial intelligence, which are based on the use of logic, rules, and symbols to represent and manipulate knowledge. These approaches can capture various aspects of human intelligence and have been since the beginning used to formalize and study problems closely related to our framework.

8.1.1 Definability

In constraint programming, the notion of *definability* [87] revolves around a dataset D and a unit $\mathbf{U}$. The task is to determine whether there exists a PPF φ that satisfies the condition $\varphi(D) = \mathbf{U}$. Under no input assumptions, its complexity is known to be coNEXP-complete. In our context, the analogous task would ask whether there exists an N$\mathbb{C}$F φ such that $inst(\varphi, \mathcal{S}) = \mathbf{U}$. When such a φ exists, it might not be a characterization. For example,

consider $\varphi_{def} = z \leftarrow \mathsf{isa}(z, \mathsf{tp}), \mathsf{located}(z, \mathsf{Florida})$. While $\varphi_{def} \longrightarrow \bar{\varphi}_a$ holds, the converse $\bar{\varphi}_a \longrightarrow \varphi_{def}$ does not. Having this in mind we can define a new computational task and study its complexity. Basically the problem DEF takes as input a unit $\mathbf{U}$ and an SKB $\mathcal{S} = (K, \varsigma)$ with $K = (D, O)$ and asks if there is no other tuple formed starting from elements belonging to the domain of D that share the same nexus of similarity shared by the tuples of the unit. Another way of looking at it, from a point of view less tied to the semantics of the problem, is to ask when and if it is the case that $ess(\mathbf{U}, \mathcal{S}) = \mathbf{U}$. We refer the reader to Section 6.3 for the notation used during the proof.

Theorem 10. DEF *is* coNEXP*-complete,* coNP*-complete and in* P *in the broad, medium and handy case, respectively. In particular, for each* $k > 0$*, LBs hold already for* DEF_k*.*

Proof. As for the upper bound from Algorithm 4, we notice that $\neg$DEF is in NEXP / NP in the broad / medium case respectively. After guessing a tuple $\tau \notin \mathbf{U}$, Algorithm 4 is executed. Line 1 runs in exponential / polynomial time, lines 2-3 run in nondeterministic exponential / polynomial time, and lines 4-5 run in exponential / polynomial time depending on the considered case.

For the handy case, instead of guessing a tuple $\tau \notin \mathbf{U}$, all possible tuples $\tau \notin \mathbf{U}$ such that $\mathbb{D}_\tau \subseteq \mathbb{D}_D$ are enumerated, where D is the dataset. The algorithm is then run for each enumerated tuple. If a tuple in the essential expansion of the unit is found, the algorithm stops and rejects; if no such tuple is found, the algorithm stops and accepts. This algorithm works in P.

The following reduction preserves our assumptions. We provide a single construction that varies only based on the input's nature. The broad case is reduced from the broad case of the starting problem, and the medium case is reduced from the medium case of the starting problem.

Regarding the lower bounds, let $+\text{ESS}_1^{\emptyset}$ be the decision problem ESS with the input unit $\mathbf{U}$ being unary, having cardinality larger than one, and the input ontology O being the empty set. We show that for each $k > 0$, $+\text{ESS}_1^{\emptyset} \leq \neg\text{DEF}_k$ via the mapping $(\mathcal{S}, \mathbf{U}, \tau) \mapsto (\mathcal{S}', \mathbf{U}^k)$, where $\mathcal{S} = (K, \varsigma)$, $K = (D, \emptyset)$, $|\mathbf{U}| > 1$, $\mathcal{S}' = (K', \varsigma')$, $K' = (D', \emptyset)$, $D' = D \cup tw(\mathbb{D}_D, k) \cup fc(\mathbb{D}_{\mathbf{U} \cup \{\tau\}})$, ς' is s.t. $\varsigma'(K', \tau') = \varsigma(K, \tau_0) \cup tw(\mathbb{D}_{\tau_0}, k) \cup fc(\mathbb{D}_{\mathbf{U} \cup \{\tau\}})$, $\tau' \in \mathbb{D}^k_{D'}$ and $\tau_0 = \mathit{off}(\tau', \mathsf{dummy})$. First let us notice that in our context given a characterization for $\mathbf{U}$ with respect to $\mathcal{S}'$, call it φ, then we can easily construct a characterization for $\mathbf{U}^k$ with respect to $\mathcal{S}'$ of the form $x_1, ..., x_k \leftarrow \bigwedge_{s \in [2...k]} \top(x_s), \mathsf{twin}_s(x_1, x_s) \wedge \bigwedge_{\alpha \in atm(\varphi)} \alpha$. Vice versa say that you have a characterization for $\mathbf{U}^k$. Then, by renaming every

occurrence of the free variables other than the first one, we will end up with a characterization for $\mathbf{U}$, having asserted this we can equivalently solve the problem by concentrating on the unary case. Let us start by assuming that $\tau \notin ess(\mathbf{U}, \mathcal{S})$ then adding the atom $\mathsf{focus}(x_1)$ to the characterization we had for $\mathbf{U}$ with respect to $\mathcal{S}$ we will end up with a formula that defines $\mathbf{U}$ with respect to $\mathcal{S}'$. This is because the atom we just added will keep away every element of $ess(\mathbf{U}, \mathcal{S})$ other than the element of $\mathbf{U}$ itself. On the other hand if $\tau \in ess(\mathbf{U}, \mathcal{S})$ then it will also hold that $\tau \in ess(\mathbf{U}, \mathcal{S}')$, this can be observed directly by constructing a characterization for the unit which will be nothing more than the previous characterization with the addition of some atoms of the form $\mathsf{focus}(y)$ together with $\mathsf{focus}(x_1)$, and again this characterization will also explain τ. What remains to be shown is that the new summary selector can be built without making complexity leaps and that its execution time turns out to be polynomial with respect to the input received, but this is evident as we only have to do a linear scan of the input, allowing us to reconstruct the initial dataset and build the new tuple. We can then relaunch the old summary selector on this new tuple. This second part runs in polynomial time by hypothesis. Additionally, we add the other elements by doing a linear scan of the input, ensuring a polynomial time complexity.

□

The decidability of definability has also been studied in symbolic logic [12], where both D and $\mathbf{U}$ are given in terms of first-order formulas over a countably infinite structure, and in graph databases [5], where path queries play the role of PPFs. In the latter case, the complexity may reach EXPSPACE. Recently, in knowledge management [20], a different variant considers a KB K instead of D, the notion of certain answers $cert(\varphi, K)$ instead of $\varphi(D)$, and looks for a (union of) conjunctive queries φ, (U)CQs for brevity (where CQs are CFs with no constant), such that $cert(\varphi, K) \odot \mathbf{U}$, where $\odot \in \{=, \subseteq, \supseteq\}$. When K is expressed in the DL-Lite$_{\mathcal{R}}$ description logic (DL) paired with a GLAV mapping, checking the existence of φ remains coNEXP-complete if $\odot \in \{=\}$, but trivializes if $\odot \in \{\subseteq, \supseteq\}$. In particular, if $\odot \in \{\supseteq\}$, then the task essentially looks for some explanation.

8.1.2 Logic Separability

In relational databases, *query by examples* (QBE) [9] considers as input a dataset D, a unit $\mathbf{U}$, and a set of negative tuples $\mathbf{U}^-$. The goal is to check whether there is a CQ φ s.t. $\varphi(D) \supseteq \mathbf{U}$ and $\varphi(D) \cap \mathbf{U}^- = \emptyset$. The complexity here may reach coNEXP. In our framework, this task would be

solved by checking whether $ess(\mathbf{U}, \mathcal{S}) \cap \mathbf{U}^- = \emptyset$. QBE has evolved into the notion of *weak separability* [35, 30, 43], where a KB K is used instead of D, and $cert(\varphi, K)$ is employed instead of $\varphi(D)$. Considered languages for K are Horn-$\mathcal{ALC}(\mathcal{I})$, $\mathcal{ALC}(\mathcal{Q})(\mathcal{I})$, GF, and FO^2 —the latter two being the guarded and the two-variables fragment of first-order logic, respectively. Studied languages for φ are (union of) CQs, $\mathcal{EL}$ or $\mathcal{ALCI}$ concepts, GF, and FO^2. The variant named *strong separability* [30, 43], asks whether there exists a formula φ such that $cert(\varphi, K) \supseteq \mathbf{U}$ and $cert(\neg\varphi, K) \supseteq \mathbf{U}^-$. Both tasks, when decidable, may reach 2EXP.

8.1.3 Reverse Engineering

Additionally, in relational databases, *reverse engineering* [80] considers as input a database D and a formula φ in some superclass of CFs that allows for comparison operators, disjunction, or aggregates. The goal is to find a formula $\varphi' \neq \varphi$ such that $\varphi(D) = \varphi'(D)$ under certain syntactic minimality conditions. Essentially, the focus is to uncover "compact" explanations that may not necessarily be characterizations, but still yield identical answers to characterizations.

8.1.4 Least General Generalizations

Finally, there are some works that show how to construct *least general generalizations* in specific settings. We are going to discuss some key approaches still using our terminology. In [23], given a dataset D over a single ternary relation *triple* encoding an RDF graph, a unary unit $\mathbf{U}$ of resources, and a characteristic function ς returning, for any resource $\langle r \rangle \in \mathbf{U}$, a set T_r of RDF triples connected to r (like our summaries which, however, are not necessarily connected), the authors show how to construct a generalized RDF graph (namely, an RDF graph with blank nodes) being connected and acting as the least common subsumer (LCS) of the resources in $\mathbf{U}$ (like our canonical characterizations). To guarantee the existence of LCSs, the authors consider only characteristic functions that return sets T_r containing at least one atom of the form $triple(r, p, o)$ for some p and o. In [44], given a KB K expressed in the $\mathcal{EL}(\mathcal{I})$ DL, a unary unit $\mathbf{U}$ with $|\mathbf{U}| = 1$, and a summary selector always returning $ent(K)$, the authors study the problem of checking whether an $\mathcal{EL}(\mathcal{I})$-characterization (called most specific concept or MSC) exists and verifying whether a given $\mathcal{EL}(\mathcal{I})$ concept is an MSC. Moreover, the authors also study the variant of these problems where the input unit contains a set of $\mathcal{EL}(\mathcal{I})$ concepts rather then entities, in order to study existence and verification of LCSs. In [78], under arbitrary (U)CQs,

KBs with an empty (onto)logical part, and summary selectors always returning the entire dataset, the authors study existence, verification, and construction of characterizations (called fitting CQs).

8.2 Topics with Subsymbolic Approaches

In this section, we will explore some of the subsymbolic approaches to artificial intelligence, which are based on the use of numerical values, vectors, and functions to model and learn from data.

8.2.1 Semantic similarity

The objective is to compute some *measure* or *distance* between pairs of entities according to *shared properties* recognizable among their *semantic connections* [29, 33]. First, the focus is on *is-a* relationships. For example, cat and tiger may have a high similarity as they are both "feline". Entities, however, can share other properties. Indeed, cat and dog both have "four paws" but are not "feline". The term *similarity*, however, is often misused to refer the broader notion of *relatedness* [17]. For example, nail and hammer are not very similar, but they are highly related. There are three main approaches to compute such measures: *string-based*, *corpus-based*, and *knowledge-based*, although finer classifications exist and some specific techniques are hybrid [17, 29]. String-based approaches operate at the level of raw char sequences [56]. For example, `pen` and `marker` would have almost nothing in common, whereas `pen` and `pan` would be considered similar! Corpus-based approaches mainly use co-occurrences [31, 47, 55, 82] or word embeddings [6, 26, 58, 88] from text. Roughly, two words are similar if they occur in similar contexts. There is hope that `pen` and `marker` are similar; however, in some cases, the same could happen with `pen` and `sheet`, which are related but not very similar. Knowledge-based approaches mainly use measures on structured resources [57, 66, 67, 74] or knowledge graph embeddings [50, 53]. They typically consider: the length of the paths between concepts; the specificity of concepts expressed, for example, as the amount of information required to describe what they are or have in common; the relatedness between words by using different kind of information such as "paths between concepts that are neither too long nor changes direction too often" or "overlaps between the textual descriptions of concepts"; the distance between learned entity embeddings.

8.2.2 Entity set expansion

The main goal of this problem is to expand a given set of entities while preserving their nexus (of similarity or of relatedness) [37, 84]. A primal form of the problem was studied by Google via a patent [79] for a system, called *Google Sets*, that "automatically creates a list from items in existing lists".[1] Starting from Audi and BMW, the system could return also Fiat, Mercedes, and so on. The topic is also known as *entity recommendation* [10] and *entity suggestion* [91]. Traditionally, the problem has been addressed by machine-learning or probabilistic models that use co-occurrence of entities within corpora to infer similarities [8, 32, 40, 55, 72, 85]. More recently, some approaches based structured resources have been proposed. They mainly use probabilistic models [18, 41, 51, 65, 74, 90] or graph embeddings [50] to determine common semantic properties.

8.2.3 Knowledge Graph Embeddings

Knowledge graph embeddings are a way of representing the entities and relations in a knowledge graph as low-dimensional vectors in a continuous space. The main goal of knowledge graph embeddings is to capture the semantic information and structure of the knowledge graph, and to enable various tasks such as link prediction, entity alignment, and knowledge graph completion. One of the challenges of knowledge graph embeddings is to measure the similarity or distance between different entities or relations, which can reflect their semantic relatedness or compatibility.

There are different approaches for learning knowledge graph embeddings for similarity, depending on how they define and optimize the similarity function. Some methods use a translational principle, which assumes that the embedding of a head entity plus the embedding of a relation should be close to the embedding of a tail entity if they form a valid triple in the knowledge graph. For example, TransE [13] uses the L1 or L2 norm to measure the distance between the embeddings, while TransH [86] and TransR [50] use a hyperplane or a relation-specific space to project the embeddings before computing the distance. Other methods use a bilinear principle, which assumes that the embedding of a head entity multiplied by a relation-specific matrix should be close to the embedding of a tail entity if they form a valid triple in the knowledge graph. For example, DistMult [89] uses a diagonal matrix to model the relation, while ComplEx [81] and RotatE [92] use complex-valued embeddings to capture the symmetry and antisymmetry of

[1] Google Sets (2003) are currently not publicly available.

relations. Some recent methods use deep neural networks to learn more expressive and flexible similarity functions for knowledge graph embeddings. For example, ConvE [25] uses a convolutional neural network to combine the embeddings of a head entity and a relation, and then computes the dot product with the embedding of a tail entity. Graph Convolutional Networks (GCN) [46] can be used to aggregate the neighborhood information of entities and relations in the knowledge graph, and then compute the similarity based on the enriched embeddings. For example, R-GCN [70] uses GCN to encode both entities and relations as node features in a relational graph, while SACN [71] uses GCN to encode only entities as node features and relations as edge features in an attributed graph.

Another line of research focuses on learning knowledge graph embeddings for similarity across multiple sources or domains, which can facilitate tasks such as entity alignment or transfer learning. For example, MTransE [19] learns cross-lingual embeddings for multilingual knowledge graphs by aligning entity pairs with translation vectors. JAPE [75] learns joint attribute-preserving embeddings for cross-lingual entity alignment by exploiting both structural and attribute information of entities. SimE-ER [76] learns similarity-based embeddings for multi-sourced knowledge graphs by using a convolutional neural network to generate similarity features of entity-pairs from their attributes, and a graph neural network to propagate the similarity features and get the final embeddings of entity-pairs.

Chapter 9

Discussion and Conclusions

In the first section of this Chapter, we will delve into the design choices that underpin our framework. These choices were particularly challenging to discuss before due to the absence of a comprehensive overview of the framework itself. Contrary to the perception of the framework as a simple amalgamation of its individual components, even a minor modification, as we will elaborate shortly, results in a profound impact.

Hence, our exploration commences with an in-depth examination of the pivotal design decisions. Each component, intricately interwoven, requires meticulous attention. The addition or removal of even a singular piece necessitates, as we will discuss, significant alterations, illustrating the complexity inherent in our framework.

Subsequently, we conduct a meticulous analysis to discern the challenges encountered during the development process. Reflecting on the past three years of research, we gain invaluable insights. These reflections illuminate not only the intricacies faced but also the invaluable lessons learned, shaping the evolution of our framework.

In the final section, we chart the course for future research endeavors. The horizons of our exploration expand as we outline the trajectories we intend to pursue. Simultaneously, we identify unresolved queries, the pursuit of which promises exciting avenues for future investigations.

9.1 Challenges and Lessons Learned

Summaries

As highlighted by previous work [23], the use of summaries is crucial when dealing with common properties. In our framework, avoiding reasonably

small summaries has several detrimental effects:

(a) *Loss of Readability:* Characterizations lose readability for humans, especially in modern knowledge graphs where all information is interconnected. Focusing on specific parts of information is essential; otherwise, the characterizations, even core types, become too large for human comprehension.

(b) *Mismatch in Similarity Nexus:* Nexus of similarity are scenario-dependent. For instance, consider a user searching for a smartphone and an advertiser creating an advertisement. The relevant information for each is distinct. Failure to focus on these differences results in irrelevant nexus of similarity, making the framework ineffective.

(c) *Computational Challenges:* The computation of the direct product of large datasets becomes computationally unfeasible, particularly if we have in mind real-time applications.

In contrast, the ability to selectively exclude irrelevant features and predicates is pivotal in generating meaningful characterizations and expansions. By concentrating on pertinent information, such as "physical aspects when comparing persons with their parents" or "skills when comparing persons with their colleagues," characterizations become concise and tailored to specific application needs.

A departure from prior work [23] lies in our insistence that summaries be closed under $\top$. For unary units, default characterizations might be assumed whenever a characterization is absent, e.g., $x \leftarrow \top(x)$. This choice becomes significant during experimentation, allowing for scenarios without atoms related to the predicate $\top$. Furthermore, this approach opens avenues for defining new languages, as explored in the next section.

However, in cases involving units of arbitrary arities, meaningful characterizations might not exist if summaries are not closed under $\top$. For example, $x, y \leftarrow r(x), \top(x), \top(y)$ would be more informative than the default characterization $x, y \leftarrow \top(x), \top(y)$, even though it would not exist without enforcing summaries to be closed under $\top$.

This strategic enforcement ensures our framework's flexibility and relevance across diverse applications and scenarios.

Explanation language

N$\mathbb{C}$Fs naturally capture shared interconnected properties because they possess unique features:

(1) *Inclusion of Constants:* NℂFs allow the use of constants, providing vital details. Without access to constants, distinctions such as "having already visited an amusement park in the United States of America" and "having already visited some place located somewhere" lose significance, highlighting the importance of informative constants.

(2) *Existential Quantification:* NℂFs support existential quantification, enabling the representation of connections beyond constants. Expressions like "somewhere in the United States of America" would be challenging without the ability to use existential quantification.

(3) *Conjunction and Joins:* Supporting conjunction and joins, NℂFs inherently express connections between entities. This feature is essential for characterizing interconnected properties.

(4) *Multiple Free Variables:* NℂFs accommodate multiple free variables, allowing the study of complex relationships, such as nexus of similarity between tuples like $\langle \mathsf{Tokyo}, \mathsf{Tokyo\ Tower} \rangle$ and $\langle \mathsf{Paris}, \mathsf{Eiffel\ Tower} \rangle$.

(5) *Avoidance of Disconnected Components:* NℂFs prevent disconnected components and disjunction, ensuring semantic connections. For instance, the absence of information like "Einstein was a scientist" in some summaries should not imply that entities share this property.

(6) *Flexibility in Connected Components and Acyclicity:* NℂFs do not enforce a single connected component or acyclicity, allowing for varied scenarios, even those with cycles. Attempting to eliminate cycles could result in loss of characterizations, making them infinitely larger but less precise.

(7) *Negation and Universal Quantification:* NℂFs waive negation and universal quantification, ensuring a focus on summary-based information. Future enhancements might include local operators serving similar functions in relation to summaries.

(8) *Absence of Built-in Equality:* NℂFs disallow built-in equality, which complicates interactions with constants.

The absence of built-in equality is particularly noteworthy. Consider the scenario where a and b are indistinguishable due to identical encounters with constants. If equality were allowed, disregarding the transposability of a and b would lead to erroneous essential expansions. In contrast, NℂFs accurately capture transposability.

Regarding the latter, consider the following SKB $\mathcal{S} = (K, \varsigma)$, where $K = (D, \emptyset)$, $D = \{r(a,a), r(a,b), r(b,a), r(b,b)\}$, $\varsigma(\langle a \rangle) = \varsigma(\langle b \rangle) = D^\top$, and $\mathbf{U} = \{\langle a \rangle\}$. Intuitively, a and b are indistinguishable as they "encounter" exactly the same constants; indeed, they are *transposable* (i.e., more than *automorphic*, a transposition is in fact a particular type of automorphism that exchanges only two objects between them leaving the role of all the others unchanged. In fact, when this happens it means that the only thing that distinguishes those two objects is their name and nothing more, because they, seen through the eyes of other objects, fulfill exactly the same role and in the same way). If we allowed equality, then $\mathit{core}(\mathbf{U}, \mathcal{S})$ would be of the form $x \leftarrow \ldots x = a$. Thus, $\mathit{ess}(\mathbf{U}, \mathcal{S}) = \mathbf{U}$, overlooking that a and b are transposable. Differently, with NℂFs we have $\mathit{ess}(\mathbf{U}, \mathcal{S}) = \{\langle a \rangle, \langle b \rangle\}$, capturing the transposability of a and b.

Additionally, NℂFs extend the concepts of rooted RDF-graphs [23] and routed CQs [43]. While related work focuses on nearly connected conjunctive formulas, NℂFs, by incorporating summaries, offer unique expressive power. When summaries differ from the entire entity set, NℂFs and other formalisms diverge in their essential expansions, highlighting the nuanced behavior introduced by summaries.

Instances of Formulas Our notion of instances of a formula is a convenient generalization of the notion of output of a formula in the presence of summaries. Indeed, whenever the given summary selector always returns $\mathit{ent}(K)^\top$, then $\mathit{inst}(\varphi, \mathcal{S}) = \varphi(\mathit{ent}(K)^\top)$. However, in general, by adopting the output $\varphi(\mathit{ent}(K)^\top)$ of a formula φ rather than their instances $\mathit{inst}(\varphi, \mathcal{S})$ in item (ii) of Definition 6, we would end up with counter intuitive definitions of explanations and characterizations. Consider, for example, the SKB $\mathcal{S} = (K, \varsigma)$, where $K = (D, \emptyset)$,

$$D = \{r_1(a_0), r_1(a_1), r_1(a_2), r_2(a_0), r_2(a_1), r_2(a_2), p(b)\},$$

$\varsigma(\langle a_0 \rangle) = \{r_1(a_0), \top(a_0)\}$, $\varsigma(\langle a_1 \rangle) = \{r_1(a_1), \top(a_1)\}$, $\varsigma(\langle a_2 \rangle) = \{\top(a_2)\}$, $\varsigma(\langle b \rangle) = D^\top$, and $\mathbf{U} = \{\langle a_0 \rangle, \langle a_1 \rangle\}$. According to $\varsigma(\langle a_0 \rangle)$ and $\varsigma(\langle a_1 \rangle)$, the expected core characterization of $\mathbf{U}$ is clearly:

$$\mathit{core}(\mathbf{U}, \mathcal{S}) \;=\; x \leftarrow r_1(x), \top(x).$$

To confirm that the latter is indeed a core, one should be able to guarantee that any other explanation for $\mathbf{U}$ can be mapped homomorphically to $\mathit{core}(\mathbf{U}, \mathcal{S})$. However, according to the definition of output of a formula, we would have that the formula $\varphi = x \leftarrow r_1(x), r_2(x), \top(x)$ is an explanation for $\mathbf{U}$ since $\varphi(\mathit{ent}(K)^\top) = \{\langle a_0 \rangle, \langle a_1 \rangle, \langle a_2 \rangle\} \supseteq \mathbf{U}$. But $\varphi \longrightarrow \mathit{core}(\mathbf{U}, \mathcal{S})$

does not hold. Thus, $core(\mathbf{U}, \mathcal{S})$ would not be a characterization according to Definition 7 and Corollary 3. Such a behaviour depends on the fact that φ contains a predicate (namely, r_2) that does not appear in the summaries of the entities of $\mathbf{U}$. By focusing on the instances, φ would not be an explanation since $\varphi(\varsigma(\langle a_0 \rangle))$ does not contain $\langle a_0 \rangle$ and, analogously, $\varphi(\varsigma(\langle a_1 \rangle))$ does not contain $\langle a_1 \rangle$.

Consider again the expected core of $\mathbf{U}$ given above, namely $core(\mathbf{U}, \mathcal{S}) = x \leftarrow r_1(x), \top(x)$. Bu using the output of a formula also in the definition of essential expansion, we would have $ess(\mathbf{U}, \mathcal{S}) = \{\langle a_0 \rangle, \langle a_1 \rangle, \langle a_2 \rangle\}$. But now, if we characterize this unit according to the given summaries, we would get the following core characterization;

$$core(ess(\mathbf{U}, \mathcal{S}), \mathcal{S}) = x \leftarrow \top(x).$$

Hence, $core(\mathbf{U}, \mathcal{S})$ would not characterize $ess(\mathbf{U}, \mathcal{S})$ any more, as desirable. Indeed, Proposition 11 is violated.

9.2 Summary and Future Works

In this thesis, we have laid out the fundamentals of a theory for establishing—in a fully semantic way—proper nexus between (tuples of) entities, and we have paved the way for future research. The theory is based on the notion of explanations, which are logical expressions that capture the semantic relations between entities in a Selective Knowledge Base (SKB). We have thus proposed a formal framework for computing and comparing explanations, in particular by finding among them special candidates called characterizations, and we have shown how those can be used to unveil the nexus of similarity between (tuples of) entities.

Summarizing we have made the following contributions:

- We have proposed a formal framework for characterizing nexus of similarity within Knowledge Bases. The framework is based in particular on the following cornerstones:

    1. Anonymous Relations, which act as fundamental units from which we want to discover the nexus of similarity.
    2. Selective Knowledge Bases, which in their essence are knowledge bases capable of adapting to the reference context through the aid of ad hoc algorithms capable of returning only relevant information for tuples of entities, this not only makes computations possible which would otherwise be too onerous, but at the same

time it gives the possibility of obtaining explanations that are highly readable by humans and strictly linked to the application branch under consideration.

3. Explanation Language, which in fact constitutes the basis for our explanations and consequently our characterizations, and which satisfies, as we have already seen, a series of desirable canons and which is very well suited to the task of defining common properties within our units.

- We have introduced our notions of explanations and characterizations as logical expressions that capture the semantic relations between (tuples of) entities in SKBs.

- We introduced a new formal object, the expansion graph, through which we showed how expansions should always, by their nature, be considered in a taxonomic and non-linear manner.

- We have defined and analyzed various decision problems linked to our framework, not with the simple intention of making it a purely stylistic exercise, but with the idea of being able to trace the frontiers of tractability and therefore be able to better look at runtime applications in the next future. In fact, each of these problems serves either to navigate the expansion graph easily or to provide a characterization that is as human readable as possible.

- We have made a first implementation of our formal framework capable not only of characterizing the nexus of similarity starting from a unary unit but also capable of solving the aforementioned decision tasks.

- Starting from the tool and the comments on the experiments carried out, we went on to further justify our design choices, making it even more evident how their role is fundamental in terms of human-oriented applications.

We have also identified some interesting directions for future research, both from the theoretical and practical viewpoints.

From the theoretical viewpoint, it will be interesting to:

- Consider further explanation languages such as classes of concepts expressed via some suitable description logic (in case of unary units), fragments of acyclic queries closed under conjunction, and queries with comparison operators between integers or with negation. These

languages could allow for more expressive and flexible explanations that can capture different aspects of the semantics of entities and their relations.

- Design novel algorithms for efficiently evaluating queries over SKBs. These algorithms could improve the scalability and performance of our framework, especially when dealing with large and complex SKBs.

From the practical viewpoint, it will be beneficial to:

- Develop prototypical Web services—implementing the proposed framework under the handy setting—on top of existing broad and well-known semantic resources such as BabelNet [60], DBpedia [49], Wikidata [83], WordNet [59] and YAGO [39]. These services could provide easy access and integration of our framework with various applications that require semantic analysis and comparison of entities.

- Identify suitable functions for defining the summaries of (tuples of) entities. We have assumed that the summaries of entities are given as input to our framework, but we have not specified how to obtain or generate them. We would like to explore different methods for defining the these summaries, such as using metadata, annotations, user feedback, or machine learning techniques.

- Devise quantitative similarity measures (such as ratings and rankings) starting from expansion graphs. We have defined similarity operators that compare sets of explanations in a qualitative way, but we have not provided any numerical values or scores that reflect the degree of nexus of similarity. We would like to develop different methods for quantifying the latter, such as using cardinality of essential expansions, entropy, distance, or similarity functions.

In summary we can say that a lot has been done but the road to removing the veil from the nexus of similarity is far from finished, and there is still a lot to discover.

Bibliography

[1] Giuseppe Agresta. Constructing and querying a selective knowledge base: A logic programming approach. In Enrico Pontelli, Stefania Costantini, Carmine Dodaro, Sarah Alice Gaggl, Roberta Calegari, Artur S. d'Avila Garcez, Francesco Fabiano, Alessandra Mileo, Alessandra Russo, and Francesca Toni, editors, *Proceedings 39th International Conference on Logic Programming, ICLP 2023, Imperial College London, UK, 9th July 2023 - 15th July 2023*, EPTCS, 2023.

[2] Giuseppe Agresta, Giovanni Amendola, Pietro Cofone, Marco Manna, and Aldo Ricioppo. Characterizing nexus of similarity between entities. *To Appear in IPS-RCRA-SPIRIT 2023*, 2023.

[3] Giovanni Amendola, Marco Manna, and Aldo Ricioppo. Characterizing nexus of similarity within knowledge bases: A logic-based framework and its computational complexity aspects. *CoRR*, abs/2303.10714, 2023.

[4] Giovanni Amendola, Marco Manna, and Aldo Ricioppo. A logic-based framework for characterizing nexus of similarity within knowledge bases. *Available at SSRN 4588766*, 2023.

[5] Timos Antonopoulos, Frank Neven, and Frédéric Servais. Definability problems for graph query languages. In Wang-Chiew Tan, Giovanna Guerrini, Barbara Catania, and Anastasios Gounaris, editors, *Joint 2013 EDBT/ICDT Conferences, ICDT '13 Proceedings, Genoa, Italy, March 18-22, 2013*, pages 141–152. ACM, 2013.

[6] Fatemeh Torabi Asr, Robert Zinkov, and Michael N. Jones. Querying word embeddings for similarity and relatedness. In *NAACL-HLT*, pages 675–684. Association for Computational Linguistics, 2018.

[7] Franz Baader, Meghyn Bienvenu, Carsten Lutz, and Frank Wolter. Query and predicate emptiness in ontology-based data access. *J. Artif. Intell. Res.*, 56:1–59, 2016.

[8] Krisztian Balog, Marc Bron, and Maarten de Rijke. Query modeling for entity search based on terms, categories, and examples. *ACM Trans. Inf. Syst.*, 29(4), 2011.

[9] Pablo Barceló and Miguel Romero. The complexity of reverse engineering problems for conjunctive queries. In Michael Benedikt and Giorgio Orsi, editors, *20th International Conference on Database Theory, ICDT 2017, March 21-24, 2017, Venice, Italy*, volume 68 of *LIPIcs*, pages 7:1–7:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[10] Roi Blanco, Berkant Barla Cambazoglu, Peter Mika, and Nicolas Torzec. Entity recommendations in web search. In Harith Alani, Lalana Kagal, Achille Fokoue, Paul Groth, Chris Biemann, Josiane Xavier Parreira, Lora Aroyo, Natasha F. Noy, Chris Welty, and Krzysztof Janowicz, editors, *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part II*, volume 8219 of *Lecture Notes in Computer Science*, pages 33–48. Springer, 2013.

[11] Manuel Bodirsky, Víctor Dalmau, Barnaby Martin, Antoine Mottet, and Michael Pinsker. Distance constraint satisfaction problems. *Inf. Comput.*, 247:87–105, 2016.

[12] Manuel Bodirsky, Michael Pinsker, and Todor Tsankov. Decidability of definability. *J. Symb. Log.*, 78(4):1036–1054, 2013.

[13] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, pages 2787–2795, 2013.

[14] Alain Bretto. *Hypergraph Theory: An Introduction*. Springer Publishing Company, Incorporated, 2013.

[15] Marco Calautti, Sergio Greco, Cristian Molinaro, and Irina Trubitsyna. Preference-based inconsistency-tolerant query answering under existential rules. *Artif. Intell.*, 312:103772, 2022.

[16] Ashok K Chandra and Philip M Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proceedings of the ninth annual ACM symposium on Theory of computing*, pages 77–90, 1977.

[17] Dhivya Chandrasekaran and Vijay Mago. Evolution of semantic similarity - A survey. *ACM Comput. Surv.*, 54(2):41:1–41:37, 2022.

[18] Jun Chen, Yueguo Chen, Xiangling Zhang, Xiaoyong Du, Ke Wang, and Ji-Rong Wen. Entity set expansion with semantic features of knowledge graphs. *J. Web Semant.*, 52-53:33–44, 2018.

[19] Muhao Chen, Yingtao Tian, Mohan Yang, and Carlo Zaniolo. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In *IJCAI*, pages 1511–1517. ijcai.org, 2017.

[20] Gianluca Cima, Federico Croce, and Maurizio Lenzerini. Query definability and its approximations in ontology-based data management. In Gianluca Demartini, Guido Zuccon, J. Shane Culpepper, Zi Huang, and Hanghang Tong, editors, *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, pages 271–280. ACM, 2021.

[21] Jill Cirasella. Google sets, google suggest, and google search history: Three more tools for the reference librarians bag of tricks. *The Reference Librarian*, 48(1), 2007.

[22] Pietro Cofone. A logic programming approach to core computation and expansion graph materialization. In Enrico Pontelli, Stefania Costantini, Carmine Dodaro, Sarah Alice Gaggl, Roberta Calegari, Artur S. d'Avila Garcez, Francesco Fabiano, Alessandra Mileo, Alessandra Russo, and Francesca Toni, editors, *Proceedings 39th International Conference on Logic Programming, ICLP 2023, Imperial College London, UK, 9th July 2023 - 15th July 2023*, EPTCS, 2023.

[23] Simona Colucci, Francesco M. Donini, Silvia Giannini, and Eugenio Di Sciascio. Defining and computing least common subsumers in RDF. *J. Web Semant.*, 39:62–80, 2016.

[24] H. Darwen. *An Introduction to Relational Database Theory.* Ventus Publishing, 2009.

[25] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *AAAI*, pages 1811–1818. AAAI Press, 2018.

[26] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[27] Ngurah Agus Sanjaya Er, Talel Abdessalem, and Stéphane Bressan. Set of t-uples expansion by example. In Gabriele Anderst-Kotsis, editor, *Proceedings of the 18th International Conference on Information Integration and Web-based Applications and Services, iiWAS 2016, Singapore, November 28-30, 2016*, pages 221–230. ACM, 2016.

[28] Ronald Fagin, Phokion G. Kolaitis, and Lucian Popa. Data exchange: getting to the core. *ACM Trans. Database Syst.*, 30(1):174–210, 2005.

[29] Yue Feng, Ebrahim Bagheri, Faezeh Ensan, and Jelena Jovanovic. The state of the art in semantic relatedness: a framework for comparison. *Knowl. Eng. Rev.*, 32, 2017.

[30] Maurice Funk, Jean Christoph Jung, Carsten Lutz, Hadrien Pulcini, and Frank Wolter. Learning description logic concepts: When can positive and negative examples be separated? In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 1682–1688. ijcai.org, 2019.

[31] Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*, 2007.

[32] Zoubin Ghahramani and Katherine A. Heller. Bayesian sets. In *NIPS*, 2005.

[33] Wael Gomaa and Aly Fahmy. A survey of text similarity approaches. *International Journal of Computer Applications*, 68(13):13–18, 04 2013.

[34] A. Grami. *Probability, Random Variables, Statistics, and Random Processes: Fundamentals & Applications*. Wiley, 2019.

[35] Víctor Gutiérrez-Basulto, Jean Christoph Jung, and Leif Sabellek. Reverse engineering queries in ontology-enriched systems: The case of expressive horn description logic ontologies. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 1847–1853. ijcai.org, 2018.

[36] J.L. Harrington. *Relational Database Design and Implementation*. Elsevier Science, 2016.

[37] Yeye He and Dong Xin. SEISA: set expansion by iterative similarity aggregation. In *WWW*, pages 427–436. ACM, 2011.

[38] Paul von Hippel. *Skewness*, pages 1340–1342. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[39] Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. YAGO2: A spatially and temporally enhanced knowledge base from wikipedia. *Artif. Intell.*, 194, 2013.

[40] Jizhou Huang, Wei Zhang, Yaming Sun, Haifeng Wang, and Ting Liu. Improving entity recommendation with search log and multi-task learning. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 4107–4114. ijcai.org, 2018.

[41] Nandish Jayaram, Arijit Khan, Chengkai Li, Xifeng Yan, and Ramez Elmasri. Querying knowledge graphs by example entity tuples. *IEEE Trans. Knowl. Data Eng.*, 27(10):2797–2811, 2015.

[42] T. Jech. *Set Theory*. Perspectives in Mathematical Logic. Springer Berlin Heidelberg, 2013.

[43] Jean Christoph Jung, Carsten Lutz, Hadrien Pulcini, and Frank Wolter. Logical separability of labeled data examples under ontologies. *Artif. Intell.*, 313:103785, 2022.

[44] Jean Christoph Jung, Carsten Lutz, and Frank Wolter. Least general generalizations in description logic: Verification and existence. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 2854–2861. AAAI Press, 2020.

[45] R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[46] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR (Poster)*. OpenReview.net, 2017.

[47] Peter Kolb. Experiments on the difference between semantic similarity and relatedness. In *NODALIDA*, 2009.

[48] Lucien Le Cam. The central limit theorem around 1935. *Statistical science*, pages 78–91, 1986.

[49] Jens Lehmann *et al.* Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2), 2015.

[50] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pages 2181–2187. AAAI Press, 2015.

[51] Matteo Lissandrini, Davide Mottin, Themis Palpanas, and Yannis Velegrakis. Graph-query suggestions for knowledge graph exploration. In Yennun Huang, Irwin King, Tie-Yan Liu, and Maarten van Steen, editors, *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, pages 2549–2555. ACM / IW3C2, 2020.

[52] Qingxia Liu, Gong Cheng, Kalpa Gunaratna, and Yuzhong Qu. Entity summarization: State of the art and future challenges. *J. Web Semant.*, 69:100647, 2021.

[53] Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. K-BERT: enabling language representation with knowledge graph. In *AAAI*, pages 2901–2908. AAAI Press, 2020.

[54] Hao Ma and Yan Ke. An introduction to entity recommendation and understanding. In Aldo Gangemi, Stefano Leonardi, and Alessandro Panconesi, editors, *Proceedings of the 24th International Conference on World Wide Web Companion, WWW 2015, Florence, Italy, May 18-22, 2015 - Companion Volume*, pages 1521–1522. ACM, 2015.

[55] Jonathan Mamou, Oren Pereg, Moshe Wasserblat, Alon Eirew, Yael Green, Shira Guskin, Peter Izsak, and Daniel Korat. Term set expansion based NLP architect by intel AI lab. In *EMNLP*, 2018.

[56] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA, 1999.

[57] Rada Mihalcea, Courtney D. Corley, and Carlo Strapparava. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI*, 2006.

[58] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *ICLR (Workshop Poster)*, 2013.

[59] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11), 1995.

[60] Roberto Navigli and Simone P. Ponzetto. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artif. Intell.*, 193, 2012.

[61] Jerzy Neyman. Outline of a theory of statistical estimation based on the classical theory of probability. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 236(767):333–380, 1937.

[62] Jerzy Neyman. On the two different aspects of the representative method: the method of stratified sampling and the method of purposive selection. In *Breakthroughs in Statistics: Methodology and Distribution*, pages 123–150. Springer, 1992.

[63] Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. Web-scale distributional similarity and entity set expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP 2009, 6-7 August 2009, Singapore, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 938–947. ACL, 2009.

[64] Christos H. Papadimitriou. *Computational complexity*. Academic Internet Publ., 2007.

[65] Alexandre Passant. dbrec - music recommendations using dbpedia. In *ISWC*, 2010.

[66] Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. Wordnet: : Similarity - measuring the relatedness of concepts. In *AAAI*, 2004.

[67] Giuseppe Pirrò. Reword: Semantic relatedness in the web of data. In *AAAI*. AAAI Press, 2012.

[68] Giuseppe Pirrò. Building relatedness explanations from knowledge graphs. *Semantic Web*, 10(6):963–990, 2019.

[69] Benjamin Rossman. Homomorphism preservation theorems. *J. ACM*, 55(3):15:1–15:53, 2008.

[70] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *ESWC*, volume 10843 of *Lecture Notes in Computer Science*, pages 593–607. Springer, 2018.

[71] Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. End-to-end structure-aware convolutional networks for knowledge base completion. In *AAAI*, pages 3060–3067. AAAI Press, 2019.

[72] Chuan Shi, Jiayu Ding, Xiaohuan Cao, Linmei Hu, Bin Wu, and Xiaoli Li. Entity set expansion in knowledge graph: a heterogeneous information network perspective. *Frontiers Comput. Sci.*, 15(1):151307, 2021.

[73] S. Staab and R. Studer. *Handbook on Ontologies*. International Handbooks on Information Systems. Springer Berlin Heidelberg, 2010.

[74] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *VLDB Endow.*, 4(11), 2011.

[75] Zequn Sun, Wei Hu, Qingheng Zhang, and Yuzhong Qu. Bootstrapping entity alignment with knowledge graph embedding. In *IJCAI*, pages 4396–4402. ijcai.org, 2018.

[76] Zhen Tan, Xiang Zhao, Yang Fang, Bin Ge, and Weidong Xiao. Knowledge graph representation via similarity-based embedding. *Sci. Program.*, 2018:6325635:1–6325635:12, 2018.

[77] Balder ten Cate and Víctor Dalmau. The product homomorphism problem and applications. In Marcelo Arenas and Martín Ugarte, editors, *18th International Conference on Database Theory, ICDT 2015, March 23-27, 2015, Brussels, Belgium*, volume 31 of *LIPIcs*, pages 161–176. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

[78] Balder ten Cate, Victor Dalmau, Maurice Funk, and Carsten Lutz. Extremal fitting problems for conjunctive queries. In Floris Geerts, Hung Q. Ngo, and Stavros Sintos, editors, *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2023, Seattle, WA, USA, June 18-23, 2023*, pages 89–98. ACM, 2023.

[79] Simon Tong and Jeff Dean. System and methods for automatically creating lists. (US Patent 7,350,187 B1), 2008.

[80] Quoc Trung Tran, Chee Yong Chan, and Srinivasan Parthasarathy. Query reverse engineering. *VLDB J.*, 23(5):721–746, 2014.

[81] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2071–2080. JMLR.org, 2016.

[82] Peter D. Turney. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *ECML*, 2001.

[83] Denny Vrandecic and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10), 2014.

[84] Richard C. Wang and William W. Cohen. Language-independent set expansion of named entities using the web. In *ICDM*, 2007.

[85] Richard C. Wang and William W. Cohen. Iterative set expansion of named entities using the web. In *ICDM*, 2008.

[86] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, pages 1112–1119. AAAI Press, 2014.

[87] Ross Willard. Testing expressibility is hard. In David Cohen, editor, *Principles and Practice of Constraint Programming - CP 2010 - 16th International Conference, CP 2010, St. Andrews, Scotland, UK, September 6-10, 2010. Proceedings*, volume 6308 of *Lecture Notes in Computer Science*, pages 9–23. Springer, 2010.

[88] Guangxu Xun, Yaliang Li, Wayne Xin Zhao, Jing Gao, and Aidong Zhang. A correlated topic model using word embeddings. In Carles Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 4207–4213. ijcai.org, 2017.

[89] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR (Poster)*, 2015.

[90] Xiangling Zhang, Yueguo Chen, Jun Chen, Xiaoyong Du, Ke Wang, and Ji-Rong Wen. Entity set expansion via knowledge graphs. In *SIGIR*, 2017.

[91] Yi Zhang, Yanghua Xiao, Seung-won Hwang, Haixun Wang, X. Sean Wang, and Wei Wang. Entity suggestion with conceptual expanation. In Carles Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 4244–4250. ijcai.org, 2017.

[92] Xiaohan Zhou, Yunhui Yi, and Geng Jia. Path-rotate: Knowledge graph embedding by relational rotation of path in complex space. In *ICCC*, pages 905–910. IEEE, 2021.