Informatics

Der Knowledge Graph Divide

Überwinden der Hürden zwischen Machine Learning, Databases und dem Semantic Web

DISSERTATION

zur Erlangung des akademischen Grades

Doktor der Technischen Wissenschaften

eingereicht von

Dipl.-Ing. Aleksandar Pavlović
Matrikelnummer 01525707

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Prof. Dr. Emanuel Sallinger

Diese Dissertation haben begutachtet:

Axel Polleres | Víctor Gutiérrez Basulto

Wien, 8. November 2024

Aleksandar Pavlović

TU WIEN Informatics

The Knowledge Graph Divide

Connecting Machine Learning, Databases, and the Semantic Web

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor der Technischen Wissenschaften

by

Dipl.-Ing. Aleksandar Pavlović
Registration Number 01525707

to the Faculty of Informatics

at the TU Wien

Advisor: Prof. Dr. Emanuel Sallinger

The dissertation has been reviewed by:

Axel Polleres

Víctor Gutiérrez Basulto

Vienna, November 8, 2024

Aleksandar Pavlović

Erklärung zur Urheberschaft

Dipl.-Ing. Aleksandar Pavlović

Hiermit erkläre ich, dass ich diese Dissertation selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe. Ich möchte darauf hinweisen, dass diese Dissertation direkt auf die Inhalte meiner Publikationen (Pavlović and Sallinger, 2023a,b; Angles et al., 2023a,b; Pavlović and Sallinger, 2024a,b,c), Bachelorarbeit (Pavlović, 2019) und Diplomarbeit (Pavlović, 2020) zurückgreift. Um die Konsistenz dieser Arbeit zu gewährleisten, wurden die Inhalte von (Pavlović and Sallinger, 2023a,b; Angles et al., 2023a,b; Pavlović and Sallinger, 2024a,b,c; Pavlović, 2019, 2020) bewusst nicht umgeschrieben, außer, wenn dies zur Anpassung der Inhalte in ein einheitliches Werk notwendig war, wie beispielsweise durch die Vereinheitlichung von Notationen oder durch das Umstrukturieren, Ergänzen und Überarbeiten von Texten. Die Inhalte meiner Bachelor- und Diplomarbeit wurden nur der Vollständigkeit halber in dieser Dissertation aufgenommen und sind daher nicht als Leistung dieser Arbeit zu betrachten. Aus diesem Grund werden Inhalte, die direkt aus meiner Bachelor- oder Diplomarbeit entstammen, in jedem Kapitel explizit erwähnt.

Wien, 8. November 2024

Aleksandar Pavlović

Declaration of Authorship

Dipl.-Ing. Aleksandar Pavlović

I hereby declare that I have written this doctoral thesis independently, that I have completely specified the utilized sources and resources, and that I have definitely marked all parts of the work - including tables, maps, and figures - which belong to other works or the internet, literally or extracted by referencing the source as borrowed. For full disclosure, I want to remark that this doctoral thesis directly reuses the content of my publications (Pavlović and Sallinger, 2023a,b; Angles et al., 2023a,b; Pavlović and Sallinger, 2024a,b,c), Bachelor's thesis (Pavlović, 2019), and Master's thesis (Pavlović, 2020). To ensure consistency, the content of (Pavlović and Sallinger, 2023a,b; Angles et al., 2023a,b; Pavlović and Sallinger, 2024a,b,c; Pavlović, 2019, 2020) was deliberately not rewritten in this dissertation, except when necessary to fit it into a consistent work by unifying notations, and restructuring, adding, and adapting texts. The content of my Bachelor's and Master's theses was only included for completeness and, thus, shall not be considered a contribution of this dissertation. Content directly reused from the Bachelor's or Master's theses is explicitly mentioned in each chapter.

Vienna, November 8, 2024

Aleksandar Pavlović

Danksagung

Wenn ich auf die Zeit meiner Promotion zurückblicke, komme ich nicht umhin, sie als eine unglaubliche Odyssee voller unerwarteter Wendungen und unvergesslicher Geschichten zu betrachten. Auf dieser Reise habe ich so viele freundliche und hilfsbereite Menschen kennengelernt, die mir dabei halfen, die Herausforderungen der akademischen Welt zu meistern. An dieser Stelle möchte ich mich bei allen, die mich auf dem Weg zur Promotion begleitet haben, herzlichen bedanken, auch wenn ich sie im Folgenden leider nicht alle namentlich erwähnen kann.

Zunächst danke ich meinem Promotionsbetreuer, Emanuel Sallinger. Er hat mich mit der wissenschaftlichen Welt bekannt gemacht und mir die Freiheit gegeben, meine Interessen ohne Vorbehalte zu erkunden. Besonders dankbar bin ich ihm für seine großzügige Unterstützung, die es mir ermöglichte, Kontakte auf Workshops und Konferenzen auf der ganzen Welt zu knüpfen. Ein großes Dankeschön geht an Reinhard Pichler, der meine Leidenschaft für die Forschung entfachte, indem er mich während meiner Bachelorarbeit in ein spannendes wissenschaftliches Projekt einbezog und den Grundstein für meine akademische Karriere legte. Ebenso dankbar bin ich unseren geschätzten Kollaborateur*innen, darunter Renzo Angles, Luigi Bellomarini und Georg Gottlob für ihre Beiträge und Anregungen während unserer gemeinsamen Projekte. Ein aufrichtiges Dankeschön richtet sich an Víctor Gutiérrez Basulto und Axel Polleres für die Zeit, die sie der Begutachtung dieser Arbeit widmeten. Ihr wertvolles und konstruktives Feedback wurde sehr geschätzt.

Ein besonderes Dankeschön geht an Ana Ozaki, die ich bei meinem ersten Sommerseminar kennenlernen durfte. Ihre Unterstützung während meines Forschungsaufenthalts in Norwegen war von unschätzbarem Wert, von der Einführung in das Description Logics Feld über die Formalisierung von Beweisideen bis hin zur Kontaktaufnahme mit ihrem umfangreichen akademischen Netzwerk. Ich habe unsere aufschlussreichen und heiteren Besprechungen sehr geschätzt, wie auch die schönen Wanderungen und Jam-Sessions in Bergen. Ich danke Bruno Figueira Lourenço herzlich für die Ermöglichung meines längsten Forschungsaufenthalts während eines Praktikums an seinem Institut in Japan. Ich bin überaus dankbar, dass er mich mit viel Geduld in die faszinierende Welt der konvexen Optimierung unterwiesen hat. Die Momente, die wir beim Wandern und Karaoke-Singen verbracht haben, werde ich in schöner Erinnerung behalten. Mein herzlicher Dank geht auch an Steven Schockaert, der stets eine Lösung parat zu haben schien, wenn wir auf

Hindernisse stießen. Seine Hilfsbereitschaft und sein geradezu ansteckender Enthusiasmus haben mich inspiriert und mir die Zuversicht gegeben, immer weiterzumachen.

Dank meiner tollen Kolleg*innen und Freund*innen bleibt mir jede Mittagspause, Konferenzreise und Freizeitaktivität in besonderer Erinnerung. Danke, Friedrich und Sarah, dass ihr meine ersten Lunch-Buddies nach der Pandemie wart und mir das Gefühl gegeben habt, zu Hause zu sein. Anni komplettierte unsere Runde bei unvergesslichen Filmabenden, Pubquizzes und NÖ-Card-Aktivitäten. Nelson, dein humorvoller und spontaner Geist brachte immer Freude in unsere Arbeit. Alex und Camillo, ihr wart die besten Vorlesungskollegen, die man sich vorstellen kann. Unsere Abenteuer in Chile, Mexiko und Costa Rica werde ich nie vergessen. Tianwei, deine organisierten chinesischen Fondues, Kinobesuche und nächtlichen Unterhaltungen haben uns einander näher gebracht. Giovanni, dein Humor, deine gute Laune und Leidenschaft für Enten waren eine ständige Quelle der Freude. Astrid, deine inspirierenden Vorträge wurden während meiner Doktorarbeit zu meinen Mantras. Ich hatte viel Freude an all unseren Aktivitäten mit Jules. Isi und Steffi, unsere häufigen Kochabende zur Verkostung internationaler Gerichte waren ein echter Spaß, auch wenn sie manchmal völlig missglückten (erinnert ihr euch noch an unsere Çiğ köfte?). Marko, danke für die unzähligen gemütlichen Spaziergänge quer durch Wien, die eine willkommene Abwechslung zum hektischen Studienalltag waren. Und Christine, Taty, Vanessa und Chrissi all die ausgefallenen und manchmal schrulligen Aktivitäten, die ihr organisiert habt, haben mir viele schöne und unvergessliche Erinnerungen beschert.

Ich bin auch der Familie meines Partners, vor allem Tanja, Rose und Jürgen, unfassbar dankbar, dass sie mir das Gefühl gegeben haben, schon immer ein Teil ihrer Familie gewesen zu sein.

Vielen Dank an meine Familie, insbesondere an meine unglaubliche Mutter Živadinka. Deine kontinuierliche Unterstützung über all die Jahre hinweg war die wichtigste Voraussetzung für meinen Weg. Ohne dich wäre dieses Studium für mich nicht möglich gewesen. Ich bin dir zutiefst dankbar für alles, was du als alleinerziehende Mutter geleistet hast, von deiner unerschöpflichen Aufopferung bis hin zu deiner ermutigenden Unterstützung. Ich danke dir aus tiefstem Herzen dafür, dass du meine größte Stütze warst.

Schließlich möchte ich mich bei meinem fantastischen Partner Max bedanken. Danke für die unzähligen Stunden, die du damit verbracht hast, meine Manuskripte korrekturzulesen, Forschungsideen anzuhören und Herausforderungen mit mir zu besprechen; ohne deine Unterstützung wäre ich nicht bis zu diesem Punkt gekommen. Von abenteuerlichen und schönen gemeinsamen Wanderungen bis hin zu Spaziergängen in unserer Nachbarschaft war jeder Moment mit dir eine wahre Freude. Unsere gemütlichen Kochabende, begleitet von Musik aus aller Welt, haben so viel Wärme und Lachen in mein Leben gebracht. Ich hätte mir keinen ermutigenderen, humorvolleren und liebevolleren Partner wünschen können. In den letzten Monaten, in denen wir gemeinsam das Verfassen unserer Diplomarbeiten und Dissertationen gemeistert haben, war ich so stolz auf das, was wir erreicht haben. Du hast meine Reise mit viel Freude und Energie begleitet. Meine Liebe zu dir geht über das hinaus, was Worte ausdrücken können (obwohl ich hier mein Bestes versucht habe). Ich freue mich auf all die Abenteuer, die uns in der Zukunft erwarten.

Acknowledgements

As I look back on my doctoral journey, I can't help but think of it as an incredible odyssey filled with unexpected twists and memorable stories. Along the way, I encountered so many kind and supportive people who helped me navigate the challenges of academia. I want to take this moment to express my heartfelt gratitude to everyone who has accompanied me on this path, even if I sadly cannot mention them all by name below.

Let me start by thanking my doctoral supervisor, Emanuel Sallinger. Your guidance introduced me to the scientific realm and provided me with the freedom to explore my interests without reservation. I am especially grateful for your generous support, which helped me connect with amazing people at workshops and conferences all over the world. A big thank you goes to Reinhard Pichler, who first ignited my passion for research, involving me in an interesting scientific project during my Bachelor's thesis and laying the foundation for my academic career. I am equally grateful to our esteemed collaborators, including Renzo Angles, Luigi Bellomarini, and Georg Gottlob for their insights and encouragement during our collaborative projects. A sincere thank you goes to Víctor Gutiérrez Basulto and Axel Polleres for the time they dedicated to reviewing this work. Your helpful and constructive feedback has been greatly appreciated.

A special mention goes to Ana Ozaki, whom I had the privilege of meeting at my first summer school. Your support during my research visit to Norway was invaluable, from teaching me description logics and formalizing proof ideas to connecting me with your vast academic network. I greatly appreciated our insightful and fun meetings, as well as the unforgettable hikes and jam sessions we enjoyed in Bergen. I sincerely thank Bruno Figueira Lourenço for enabling my longest research stay during an internship at his institute in Japan. You introduced me to the fascinating world of convex optimization, and I truly appreciated your patience during our interdisciplinary work. The moments we shared hiking and singing karaoke are memories I'll hold close. My heartfelt thanks go to Steven Schockaert, who always seemed to have a solution ready when we hit roadblocks. Your support and almost contagious enthusiasm inspired me and gave me the confidence to keep pushing forward.

To my amazing fellow doctoral colleagues and friends, you made every lunch break, conference trip, and leisure activity memorable. Thank you, Friedrich and Sarah, for being my first lunch buddies post-pandemic, making me feel right at home. Anni completed our group with unforgettable movie nights, pub quizzes, and "NÖ-Card"

activities. Nelson, your playful and spontaneous spirit always brought joy to our days. Alex and Camillo, you were the best lecture colleagues one could imagine; our adventures in Chile, Mexico, and Costa Rica will remain some of my most treasured memories. Tianwei, your organization of delightful hot-pot dinners, cinema visits, and late-night entertainments brought us closer together. Giovanni, your humor, good vibes, and passion for ducks were a constant source of joy. Astrid, your inspirational talks became mantras for me during my PhD; I loved all our fun activities with Jules. To Isi and Steffi, our frequent cooking sessions exploring global cuisines were a total blast, even when sometimes totally unsuccessful (remember our Çiğ köfte?). Marko, thank you for the numerous leisurely walks we had all across Vienna; they were a welcome respite from the hustle of academic life. And Christine, Taty, Vanessa, and Chrissi, all the fun and quirky activities you organized created many special memories I'll treasure forever.

I am also incredibly grateful to my partner's family, particularly Tanja, Rose, and Jürgen, for making me feel like I have always been part of their family.

Thank you to my family, especially my incredible mother, Živadinka. Your continuous support over the years has been the principal foundation of my journey, and without you, pursuing higher education would not have been possible. I am profoundly grateful for everything you have done as a single mother, from your sacrifices to your encouragement. Thank you from the bottom of my heart for being my greatest champion.

Finally, I want to thank my fantastic partner, Max. Thank you for the countless hours proofreading my manuscripts, listening to my research ideas, and discussing faced challenges; without your support, I wouldn't have reached this point. From sharing daring yet beautiful hikes to strolling around our neighborhood, every moment with you has been a joy. Our cozy cooking sessions, accompanied by music from around the world, have brought so much warmth and laughter into my life. I could not ask for a more encouraging, entertaining, and loving partner. These past few months, as we've navigated the challenges of writing our theses and dissertations together, I have felt an immense pride in what we've accomplished. You have filled my journey with both joy and strength, and my love for you goes beyond what words can express (although I've tried my best here). I look forward to all the adventures that await us in the future.

Kurzfassung

In den letzten zehn Jahren haben Knowledge Graphs (KGs) ein enormes Interesse seitens der Industrie und Wissenschaft geweckt. Dabei gibt es drei große Forschungsbereiche, Machine Learning (ML), Database (DB) und Semantic Web (SW), die an der Repräsentation und dem Management von KGs arbeiten. Jedoch bestehen große Diskrepanzen innerhalb der Forschung an KGs, welche diese Arbeit folgend identifiziert und überwindet:

Inferenzproblem. KGs sind von Natur aus unvollständig. Aus diesem Grund wurden ML-basierte Knowledge Graph Embedding Models (KGEs) erforscht, welche vielversprechende Ergebnisse für die Vorhersage fehlender Beziehungen liefern. Weiters, werden im DB- und SW-Bereich Eigenschaften von Daten mithilfe von logische Regeln modelliert. Allerdings können grundlegende Regeln von bestehenden KGEs nicht erfasst werden, d. h. Vorhersagen unter Einhaltung dieser Regeln treffen. Insbesondere das Erfassen von (*i*) General-Composition- und (*ii*) Composition- und Hierarchy-Regeln sind zentrale Hindernisse, die es zu bewältigen gilt. Diesen Herausforderungen entgegen stellen wir unser ExpressivE-Modell, welches Entitätspaare als Punkte und Beziehungstypen als Hyperparallelogramme im virtuellen Tripelraum $\mathbb{R}^{2d}$ darstellt. Dieses Modelldesign erlaubt es ExpressivE, eine Vielzahl von logischen Regeln zu erfassen und bietet zugleich eine intuitive und konsistente geometrische Interpretation der Parameter und erfassten Regeln von ExpressivE.

Skalierbarkeitsproblem. Außerdem stellen die SW- und DB-Bereiche riesige KGs bereit, die effiziente KGEs erfordern. Für gute Vorhersageergebnisse benötigen die meisten ML-basierten KGEs jedoch hochdimensionale oder komplexe Vektorräume zur Abbildung von KGs, wodurch ihr Speicher- und Rechenzeitbedarf drastisch ansteigt. Daher stellt die Entwicklung effizienter KGEs ein weiteres zentrales Hindernis dar, welches die Felder ML, DB und SW voneinander trennt. Angesichts dieser Herausforderung stellen wir SpeedE vor, ein euklidisches KGE, das (*i*) über ausgeprägte Inferenzfähigkeiten verfügt, (*ii*) mit KGEs auf dem Stand der Technik konkurrenzfähig ist und diese auf den Benchmarks YAGO3-10 und WN18RR signifikant übertrifft, während (*iii*) es auf WN18RR bei vergleichbarer Performance für die Vorhersage von fehlenden Beziehungen lediglich ein Fünftel der Trainingszeit und ein Viertel der Parameter des ExpressivE Modells benötigt.

Datenverwaltungsproblem. Die klassische KG-Forschung wurde vor allem innerhalb der DB- und SW-Bereichen vorangetrieben. Dennoch besteht eine gewisse Differenz zwischen den Ansätzen aus diesen beiden Forschungsfeldern. Während beispielsweise

Sprachen wie SQL oder Datalog im DB-Bereich weit verbreitet sind, werden im SW-Bereich ganz andere Sprachen wie SPARQL und OWL verwendet. Dies erschwert jedoch die Integration von KGs aus beiden Forschungsfeldern, weshalb die Kompatibilität zwischen DB- und SW-Technologien eine dringende Herausforderung darstellt. Folglich stellen wir das SparqLog-System vor, ein einheitliches und konsistentes KG-Management-Framework, welches wichtige Anforderungen des SW- und des DB-Bereichs erfüllt.

Abstract

Over the past decade, Knowledge Graphs (KGs) have received enormous interest from industry and academia. However, there are three key research communities, namely the Machine Learning (ML), Database (DB), and Semantic Web (SW) communities, studying KGs with major gaps between them. This dissertation is about bridging their divisions:

Reasoning Divide. KGs are inherently incomplete. Therefore, the ML community has proposed Knowledge Graph Embedding Models (KGEs), achieving promising results for predicting missing links. Key data properties in the DB and SW fields are typically represented via logical rules. However, any current KGE cannot capture vital rules, i.e., infer missing links while adhering to such rules. Capturing (*i*) general composition and (*ii*) composition and hierarchy rules jointly are crucial open problems. To bridge this division, we introduce the ExpressivE model that embeds pairs of entities as points and relations as hyper-parallelograms in the virtual triple space $\mathbb{R}^{2d}$. This model design allows ExpressivE to capture a rich set of logical rules while offering an intuitive and consistent geometric interpretation of ExpressivE embeddings and their captured rules.

Scalability Divide. Even more, the SW and DB communities provide massive KGs, calling for efficient KGEs. However, most contemporary ML-based KGEs require high-dimensional embeddings or complex embedding spaces for competitive prediction results, drastically raising their space and time requirements. Thus, developing efficient KGEs makes up another central open problem dividing the SW, DB, and ML fields. Facing this challenge, we propose SpeedE, a Euclidean KGE that (*i*) has strong inference capabilities, (*ii*) is competitive with state-of-the-art KGEs, significantly outperforming them on the YAGO3-10 and WN18RR benchmarks, and (*iii*) dramatically increases their efficiency, needing on WN18RR solely a fifth of the training time and a fourth of the parameters of the best-performing model (ExpressivE) to reach the same link prediction performance.

Data Management Divide. Above all, the DB and SW communities have driven classical KG research. However, there remains a divide between approaches from these two fields. For instance, while languages such as SQL or Datalog are widely used in the DB area, a vastly different set of languages, such as SPARQL and OWL, is used in the SW area. This mismatch, however, makes blending KGs from both communities a complex endeavor, rendering the interoperability between DB and SW technologies a pressing open challenge. Thus, we present the SparqLog system, a uniform and consistent KG management framework meeting essential requirements from the SW and DB fields.

Contents

CHAPTER 1

Introduction

Since Google launched its first Knowledge Graph (KG), we have seen intensive work on this topic from both industry and academia. For instance, KGs such as Freebase (Bollacker et al., 2007) and WordNet (Miller, 1995) lie at the heart of numerous applications such as recommendation (Cao et al., 2019), question answering (Zhang et al., 2018), information retrieval (Dietz et al., 2018), and natural language processing (Chen and Zaniolo, 2017).

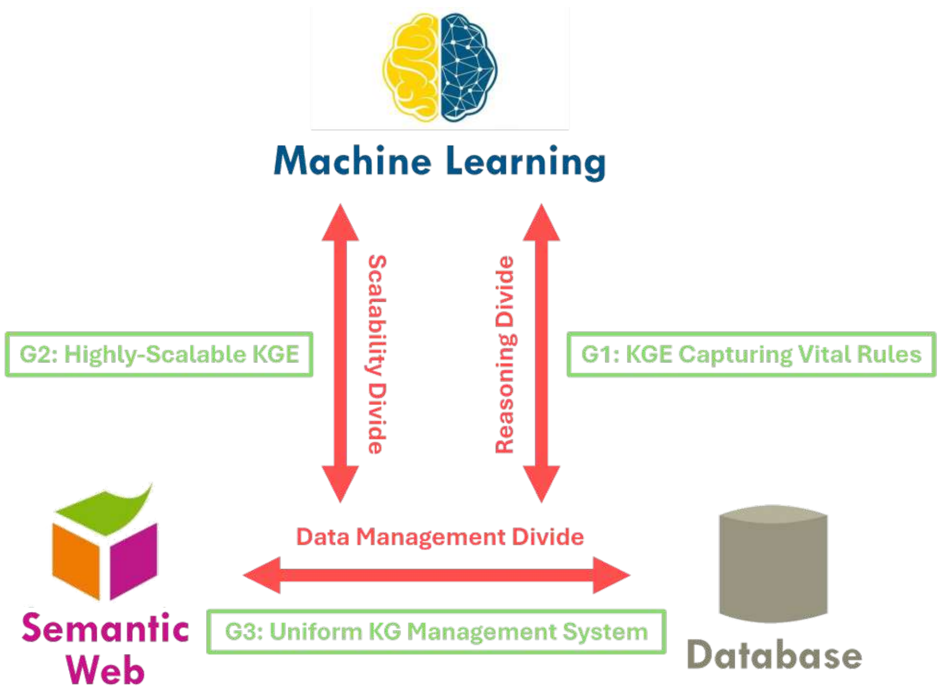

Figure 1.1: Gaps between KG research and derived research goals. The symbols for the ML[1], DB[2], and SW[3] communities are taken from the sources listed in the footnotes.

However, as visualized in Figure 1.1, three research communities are working on the representation and management of KGs with major gaps between them, namely the *Machine Learning (ML)*, *Database (DB)*, and *Semantic Web (SW)* communities. In this dissertation, we study the vast landscape of KG research, finding that it is divided along at least three dimensions, which we named the reasoning, scalability, and data management divide. The goal of this work is to overcome these separating walls by breaking down the identified divisions between these communities. Thus, the following sections discuss each of the dimensions separating KG research, subsequently deriving research goals that express favorable properties of potential solutions.

1.1 Reasoning Divide

One of the critical challenges of the intersection of the ML, DB, and SW communities is to bring together *machine learning* models and – typically logic-based – *data management* approaches. This challenge is especially apparent in the field of *graph data management* since KGs are typically highly incomplete (West et al., 2014):

On the one hand, the ML community has directed substantial research toward approaches for Knowledge Graph Completion (KGC) (Wang et al., 2017), i.e., predicting missing links from the data stored in the KG. The introduction of Knowledge Graph Embedding Models (KGEs) has yielded promising results for KGC by (*i*) representing entities and relations of a KG as embeddings in the semantic vector space and (*ii*) computing scores over these embeddings to quantify the plausibility of missing links (Wang et al., 2017). On the other hand, the DB and SW communities typically represent major data properties through *constraints*, *axioms*, or *dependencies* expressed as *logical rules*.

Logical Rule	ExpressivE	BoxE	RotatE	TransE	DistMult	ComplEx
Symmetry: $r_1(X,Y) \Rightarrow r_1(Y,X)$	✓	✓	✓	✗	✓	✓
Anti-symmetry: $r_1(X,Y) \land r_1(Y,X) \Rightarrow \bot$	✓	✓	✓	✓	✗	✓
Inversion: $r_1(X,Y) \Leftrightarrow r_2(Y,X)$	✓	✓	✓	✓	✗	✓
Comp. def.: $r_1(X,Y) \land r_2(Y,Z) \Leftrightarrow r_3(X,Z)$	✓	✗	✓	✓	✗	✗
Gen. comp.: $r_1(X,Y) \land r_2(Y,Z) \Rightarrow r_3(X,Z)$	✓	✗	✗	✗	✗	✗
Hierarchy: $r_1(X,Y) \Rightarrow r_2(X,Y)$	✓	✓	✗	✗	✓	✓
Intersection: $r_1(X,Y) \land r_2(X,Y) \Rightarrow r_3(X,Y)$	✓	✓	✓	✓	✗	✗
Mutual exclusion: $r_1(X,Y) \land r_2(X,Y) \Rightarrow \bot$	✓	✓	✓	✓	✓	✓

Table 1.1: This table lists logical rules that several KGEs can capture, where ✓ represents that the KGE can capture the rule and ✗ that it cannot capture the rule. We have proposed the ExpressivE model in (Pavlović and Sallinger, 2023b).

However, there is a substantial challenge in this: Many KGEs cannot respect vital logical rules – termed *capturing* rules – which describe a KGE's ability to infer missing

[1]https://ml.studentorg.berkeley.edu/ (last visited 07/25/2024)
[2]https://www.dbai.tuwien.ac.at/ (last visited 07/25/2024)
[3]https://www.w3.org/2007/10/sw-logos.html (last visited 07/25/2024)

triples while adhering to such logical rules. The *composition* of relations is a fundamental constraint or dependency for data management – even more so in graph data management, where it allows the description of paths. Recently, however, it was discovered that existing KGEs only capture a fairly limited notion of composition (Zhang et al., 2019; Abboud et al., 2020; Lu and Hu, 2020; Gao et al., 2020), solely capturing *compositional definition*, not *general composition* (see Table 1.1 for the defining formulas). Even more, while existing KGEs capture hierarchy (Yang et al., 2015a; Kazemi and Poole, 2018; Trouillon et al., 2016; Abboud et al., 2020) and compositional definition (Bordes et al., 2013; Sun et al., 2019; Zhang et al., 2019; Lu and Hu, 2020) individually, they cannot capture both rules simultaneously (see Table 1.1).

While the extensive research on composition (Bordes et al., 2013; Sun et al., 2019; Zhang et al., 2019; Lu and Hu, 2020) and hierarchy (Yang et al., 2015a; Trouillon et al., 2016; Kazemi and Poole, 2018; Abboud et al., 2020) highlights their importance, any KGE so far is incapable of (*i*) capturing general composition, (*ii*) capturing composition and hierarchy jointly, and (*iii*) providing a geometric interpretation of captured rules.

[G1] Thus, a crucial open challenge for the ML, DB, and SW communities is to overcome these limitations by introducing a KGE that captures a wide range of logical rules relevant to both the DB and SW communities, specifically the core inference rules (shown in Table 1.1 and discussed in Chapter 2).

1.2 Scalability Divide

The SW and DB communities provide massive KGs, containing millions of links (Mahdisoltani et al., 2015), which calls for scalable and efficient KGEs. However, most contemporary KGEs developed by the ML community suffer from efficiency problems.

On the one hand, contemporary KGEs explored increasingly *more complex* embedding spaces to boost their KGC performance (Sun et al., 2019; Zhang et al., 2019; Cao et al., 2021). However, more complex embedding spaces typically require more costly operations, leading to a lower time efficiency than Euclidean KGEs (Wang et al., 2021).

On the other hand, most KGEs require *high-dimensional embeddings* to reach state-of-the-art KGC performance, leading to increased time and space requirements (Chami et al., 2020; Wang et al., 2021). Thus, the need for (*i*) complex embedding spaces and (*ii*) high-dimensional embeddings lowers the efficiency of KGEs and, thus, limits their scalability. In addition, these needs hinder the application of KGEs in resource-constrained environments, especially in mobile smart devices (Sun et al., 2019; Zhang et al., 2019; Wang et al., 2021).

Although there has been much work on scalable KGEs, any such work has focused exclusively on either reducing the embedding dimensionality (Balazevic et al., 2019a; Chami et al., 2020; Bai et al., 2021) or using simpler embedding spaces (Kazemi and Poole, 2018; Zhang et al., 2020; Pavlović and Sallinger, 2023b), thus addressing only one side of the efficiency problem.

[G2] As considerable time and space requirements hinder the application of KGEs on massive KGs with millions of links, another crucial open challenge for the ML, DB, and SW communities is to overcome these limitations by designing a highly resource-efficient KGE that reaches state-of-the-art performance under low embedding dimensionalities while utilizing the Euclidean embedding space.

1.3 Data Management Divide

Above all, classical KG research has been driven by the DB community and the SW community. However, there still remains a certain divide between the KG management systems proposed by these two communities. For instance, while languages such as the Structured Query Language (SQL) or Datalog are widely used in the DB area for modeling and querying databases, a vastly different set of languages, such as the SPARQL Protocol and RDF Query Language (SPARQL) and the Web Ontology Language (OWL), is used in the SW area. This mismatch, however, makes blending KGs from both communities a complex endeavor, rendering the interoperability between DB and SW technologies a pressing open challenge. To tackle this challenge, we identified a set of criteria that the DB and SW communities expect from a KG management system.

Of major importance to the *SW community* is the compliance with the standards of the World Wide Web Consortium (W3C):

- **(R1) SPARQL Feature Coverage**. The query language SPARQL is one of the major Semantic Web standards. Therefore, we require the support of the most commonly used SPARQL features.
- **(R2) Bag Semantics**. SPARQL employs per default *bag semantics* (also referred to as *multiset semantics*) unless specified otherwise in a query. We therefore require the support of this.
- **(R3) Ontological Reasoning**. OWL 2 QL to support ontological reasoning is a major Semantic Web standard. Technically, for rule-based languages, this means that existential quantification (i.e., “object invention”) in the rule heads is required.

The *DB community* puts particular emphasis on the expressive power and efficient evaluation of query languages. This leads us to the following additional requirement:

- **(R4) Full Recursion**. Full recursion is vital to provide the expressive power needed to support complex querying in business applications and sciences (see, e.g., (Przymus et al., 2010)), and it is the main feature of the relational query language Datalog (Vianu, 2021). Starting with SQL-99, recursion has also been integrated into the SQL standard, and most relational database management systems have meanwhile incorporated recursion capabilities to increase their expressive power.

Finally, for an approach to be *accepted and used in practice*, we formulate the following requirement for both communities:

- **(R5) Implemented System**. Both communities require an implemented system. This makes it possible to verify if the theoretical results are applicable in practice and to evaluate the usefulness of the approach in real-world settings.

The above-listed requirements explain why there exists a certain gap between the SW and DB communities. As seen in Table 1.2 and discussed in detail in Chapter 3, several attempts have been made to close this gap. However, no approach has fulfilled both sides' requirements so far. Indeed, while existing solutions individually satisfy some of the requirements listed above, all of them fail to meet other central requirements.

Requirement	SPARQL (R1)	Bag Semantics (R2)	OWL 2 QL (R3)	Full Recursion (R4)	Implementation (R5)
DLVhex-SPARQL Plugin (Polleres and Schindlauer, 2007)	✗	✓	✗	✓	✓
Translation of SPARQL 1.1 to DLV (Polleres and Wallner, 2013)	✓	✓	✗	✓	✗
Analysis of SPARQL Bag Semantics (Angles and Gutierrez, 2016a)	✗	✓	✗	✓	✗
Vadalog System (Bellomarini et al., 2018)	✗	✗	✓	✓	✓
Warded Datalog$^{\pm}$ with Bag Semantics (Bertossi et al., 2019)	✗	✓	✓	✓	✗
SparqLog System (Angles et al., 2023b,a)	✓	✓	✓	✓	✓

Table 1.2: As in (Pavlović, 2020), this table lists the requirements that several relevant KG management systems satisfy, where ✓ depicts that the requirement is satisfied and ✗ that it is not satisfied. We have proposed the SparqLog system in (Angles et al., 2023a,b).

[G3] Thus, a pressing open challenge for the DB and SW communities is to develop one uniform and consistent KG management system that satisfies the requirements of both communities (R1–R5).

1.4 Methodology

This dissertation employs a very diverse set of methodologies, sampling appropriate methods from various areas of Computer Science. In particular, each of the Chapters starts by (*i*) using formal methods to introduce a theoretical model and prove interesting properties, (*ii*) implementing a system (and optimizing it), bridging theory and practice, and (*iii*) designing experiments for empirically evaluating the implemented system. We will consider each of these methodologies in more detail in the following paragraphs.

Formal Methods. We use formal methods to study and, subsequently, show the properties of ML models and DB/SW systems. In particular, we employ formal methods to (*i*) design theoretical concepts for these models/systems. Subsequently, we use these theoretical concepts to prove interesting properties. In the case of our proposed KGEs, we study properties such as (*ii*) which types of graphs a given KGE can represent, (*iii*) which logical rules a KGE can capture, and (*iv*) a KGE's space requirements and number of computational steps necessary for predicting new links during inference. In the case of our proposed KG management system, we design a translation from SPARQL — the standard querying language of the SW — to a Datalog dialect — an important language family for the DB community — and prove (*v*) the correctness of this translation.

Implementation. Based on the theoretical concepts introduced and their proven properties, we implement our solutions for further evaluation. For this, we have to overcome obstacles separating theory and practice. For instance, in the case of our proposed KGEs, we need to (*i*) parameterize the models in a memory-efficient way and (*ii*) design loss functions that are suitable for the considered optimization task. In the case of our proposed KG management system, we need to consider (*iii*) technical details for correctly answering SPARQL queries in practice, such as the different reasoning modes of the chosen Datalog-based system that we translate SPARQL to. Furthermore, we make our implemented KGEs and KG management system publicly available on GitHub to allow for their comfortable reuse.

Empirical Analyzes. Finally, we empirically investigate the practical behavior of our implemented KGEs and KG management system. In particular, we (*i*) review state-of-the-art approaches and standard benchmarks for various tasks such as KGC and query answering, (*ii*) benchmark our implementations to compare their performance to state-of-the-art systems, and (*iii*) study further properties empirically, such as their space and time efficiency. In the case of our proposed KGEs, we (*iv*) design experiments to test hypotheses, providing further explanations for the performance boosts of our KGEs. For instance, we study the importance of different components of our KGEs by conducting ablation studies. In the case of our proposed KG management system, we additionally (*v*) investigate the compliance of the system's query results to the SPARQL standard, providing empirical evidence that our KG management system also answers queries correctly in practice.

1.5 Publications

This dissertation is based on our following papers:

- Expressive: A Spatio-Functional Embedding for Knowledge Graph Completion.
  Aleksandar Pavlović and Emanuel Sallinger.
  ICLR 2023.

- SpeedE: Euclidean Geometric Knowledge Graph Embedding Strikes Back.
  Aleksandar Pavlović and Emanuel Sallinger.
  NAACL 2024.

- SparqLog: A System For Efficient Evaluation of SPARQL 1.1 Queries via Datalog.
  Renzo Angles, Georg Gottlob, Aleksandar Pavlović, Reinhard Pichler, and Emanuel Sallinger.
  VLDB 2024.

  and the full version of this paper (including its appendix) is available at

  SparqLog: A System for Efficient Evaluation of SPARQL 1.1 Queries via Datalog [Experiment, Analysis and Benchmark].
  Renzo Angles, Georg Gottlob, Aleksandar Pavlović, Reinhard Pichler, and Emanuel Sallinger.
  CoRR, 2023.

- Raising the Efficiency of Knowledge Graph Embeddings While Respecting Logical Rules (Short Paper).
  Aleksandar Pavlović and Emanuel Sallinger.
  AMW 2024.

- Expressive and Geometrically Interpretable Knowledge Graph Embedding (Extended Abstract).
  Aleksandar Pavlović and Emanuel Sallinger.
  AIROV 2024.

- Building Bridges: Knowledge Graph Embeddings Respecting Logical Rules (Short Paper).
  Aleksandar Pavlović and Emanuel Sallinger.
  AMW 2023.

This dissertation directly incorporates content from these papers, such as tables, figures, definitions, theorems, proofs, and text. Specifically, this chapter reuses content from each of the listed publications (Angles et al., 2023a,b; Pavlović and Sallinger, 2023a,b, 2024a,b,c). Also, Section 1.3, the preliminaries introduced in Chapter 2, the related work reviewed in Chapter 3, and for completeness also Chapter 6 directly reuse content from my Bachelor's (Pavlović, 2019) and Master's theses (Pavlović, 2020) without rewriting it for consistency. Section 6.1 explicitly discusses which parts of Chapter 6 are from my Bachelor's and Master's theses and which are novel to state the contributions of this dissertation clearly. Furthermore, at the start of each of the following chapters, I highlight the specific papers from which material is incorporated.

1.6 Use Case

Let us now consider a use case in the domain of financial KGs, where all three research goals G1–G3 come together. One particularly interesting challenge for central banks lies in the estimation of the *collateral eligibility* of companies, known in the literature also as the asset eligibility or *close link* problem (Atzeni et al., 2020). Basically, this problem is concerned with estimating the risk of giving a certain loan to a company x secured by collateral supplied by another company y. Regulations from the European Central Bank[1] state that a company y cannot be the guarantor for a company x if it is too closely connected to it with regard to ownership structure. The regulation defines a pair of closely linked entities in detail, including, for instance, the case that the counterpart x and its guarantor company y cannot be owned by a single third party entity that owns 20% or more of the shares of both of them. In addition to ownership structures, family links play an essential role in close link detection, as family businesses often act as a single center of interest and should be treated as such (Atzeni et al., 2020). Thus, Atzeni et al. (2020) extend the briefly mentioned close link definition of the European Central Bank based on family links by considering that, given two family members z_1 and z_2, z_1 cannot own 20% or more of the shares of the counterpart x when z_2 already owns 20% or more of the shares of its guarantor company y and vice versa. We will discuss Atzeni et al. (2020)'s close link definition later in greater detail.

Challenges in Close Link Detection. The Central Bank of Italy curates a large financial KG for many regulatory purposes, including the detection of close links (Atzeni et al., 2020; Baldazzi et al., 2022). There are many challenges that central banks have to face in order to identify these links, as (*i*) they are often hidden inside intricate structures of company ownership relations, requiring complex recursive rules to reason about the KG's data (Baldazzi et al., 2022); (*ii*) family relations may also play a vital role in close link detection (Atzeni et al., 2020), whose information is not readily available to the central bank but needs to be retrieved and incorporated from other sources such as the SW; (*iii*) the information on family relations is highly incomplete, thus missing links need to be predicted; and (*iv*) the financial KG is very large, requiring efficient systems for predicting missing links, querying, and reasoning over KGs. In the following paragraphs, we will see how this dissertation's contributions interplay to address these challenges. First, we will discuss some critical financial notions that form the foundation of this use case.

Direct Ownership. As briefly mentioned, the financial KG from the Central Bank of Italy contains, among other data, ownership relations between shareholders and companies (Atzeni et al., 2020). An example of such a graph is depicted in Figure 1.2, where $P1$ and $P2$ represent shareholders and $C1$–$C5$ companies. The labels of links denoted with black solid arrows represent the percentage of a company's shares directly owned by an entity. For instance, Figure 1.2's graph displays a scenario in which $P1$ directly owns 30% of company $C1$ and 40% of company $C3$.

Total Ownership. Next to direct ownership, Atzeni et al. (2020) define the total ownership of a company y held by an entity x as follows. Let $p_{x,y}$ depict all m simple paths over ownership links from x to y, where a simple path between x and y is denoted as $p^i_{x,y}$ with $1 \leq i \leq m$ and $i \in \mathbb{N}$. Furthermore, let Ω be the set of direct ownership links and $f_s : \Omega \to (0, 1]$ be the function that maps a link to its corresponding

[1] https://www.ecb.europa.eu/ecb/legal/pdf/l_33120111214en000100951.pdf (last visited 01/11/2024)

weight. Then the total ownership of a company y held by an entity x is defined as $Own_{total}(x, y) := \sum_{p^i_{x,y} \in p_{x,y}} \prod_{\omega \in p^i_{x,y}} f_s(\omega)$. For instance, in Figure 1.2, we can see that $C1$ owns 10% of $C2$ directly and 15% of $C2$ over the path $C1 \rightarrow C4 \rightarrow C2$ indirectly. Thus, $C1$ owns 25% of $C2$ in total.

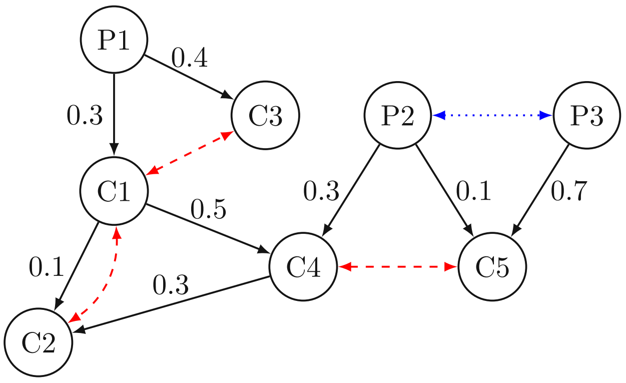

Figure 1.2: Financial KG containing ownership links (black solid arrows), which is extended with family links (blue dotted arrow) through embedding-based predictions and close links (red dashed arrows) through rule-based reasoning. This figure is based on Atzeni et al. (2020)'s examples of financial KGs.

Close Links. Based on total ownership and family relations, Baldazzi et al. (2022); Atzeni et al. (2020) define a close link between two entities x and y as follows. Two entities x and y are in a close link relationship if: (1) the total ownership of y held by x is at least 20% of the equity of y or respectively if the total ownership of x held by y is at least 20% of the equity of x; or (2) a third entity z exists that owns in total at least 20% of the equity of both x and y; or (3) two entities z_1 and z_2 with $z_1 \neq z_2$ exist that are in a family relationship, where z_1 (respectively z_2) owns in total at least 20% of x's equity and z_2 (respectively z_1) owns in total at least 20% of y's equity. For example, in Figure 1.2, (i) $C1$ and $C2$ have a close link relationship by Point 1, as $C1$ holds in total more than 20% of direct and indirect shares of $C2$; (ii) $C1$ and $C3$ are in a close link relationship by Point 2 as $P1$ holds more than 20% of the shares of $C1$ and $C3$; and (iii) assume that $P2$ and $P3$ are in a family relation (e.g., they are siblings) then $C4$ and $C5$ are in a close link relationship by Point 3, as $P2$ holds more than 20% of $C4$'s equity and $P2$'s family member $P3$ holds more than 20% of the shares of $C5$.

Data Management. With all these definitions in place, we can now examine how the combination of this dissertation's contributions tackles the challenges of this use case. On the one hand, representing the total ownership relation with an ontology requires complex recursive rules together with aggregation, demanding for the power of Datalog-based KG management solutions, in particular, their capability to express full recursion (Baldazzi et al., 2022). On the other hand, family links are typically not contained in financial KGs of central banks (Atzeni et al., 2020). Thus, the integration of other data sources is necessary. Large KGs from the SW, such as DBpedia (Auer et al.,

2007) and Wikidata (Vrandecic and Krötzsch, 2014), hold massive amounts of publicly available knowledge, such as family connections of famous entrepreneurs. Even more, they provide ontologies over family relations that allow to reason and extend the already provided data. Furthermore, the SW community defines a standard querying language for KGs (SPARQL) that is widely accepted and quite expressive, allowing the retrieval of information about recursive scenarios in KGs with concise queries. For these reasons, this use case demands naturally a KG management system that blends ontological querying in SPARQL with the rule-based inference capabilities of Datalog-based engines (in particular full recursion), such as our SparqLog system (see Chapter 6). Even more, SparqLog is highly competitive in terms of efficiency to state-of-the-art SW querying and reasoning systems while not being solely a SPARQL engine but a uniform and consistent framework for querying and reasoning over KGs of the DB and SW world. We will discuss the SparqLog system and its capabilities in greater detail in Chapter 6.

Predicting Missing Family Links. Now that we have seen how SparqLog solves the problem of representing, querying, and reasoning over close links, we are still left with another problem. In particular, while we can retrieve some family relations for famous entrepreneurs from public data sources, most family relations remain hidden. This challenge calls for KGEs to predict missing family relations based on the known ones and ownership structures. However, contemporary KGEs are not suitable for this task, as they cannot learn vital logical rules jointly. In particular, many properties of family relations can be naturally described by the core inference rules, requiring specifically both hierarchy and general composition rules. For instance, a KGE for predicting missing family links should be able to learn to infer links based on the hierarchy rule $\mathit{mother\_of}(X, Y) \Rightarrow \mathit{parent\_of}(X, Y)$, expressing that a mother is also a parent; and the general composition rule $\mathit{mother\_of}(X, Y) \wedge \mathit{parent\_of}(Y, Z) \Rightarrow \mathit{grandparent\_of}(X, Z)$, stating that the mother of a parent is a grandparent. We closed this gap by developing ExpressivE, the first KGE that captures all core inference rules, allowing it to capture many key properties of family relations (Chapter 4). Finally, since the financial graph of the Central Bank of Italy is very large, scalable solutions for predicting missing links are required. As most KGEs suffer from at least one efficiency problem (see Section 1.2), limiting their scalability, new frontiers of resource-efficient KGE research are required. We solved this scalability issue by proposing the highly efficient SpeedE model that lowers the training and inference time of state-of-the-art KGEs dramatically while preserving ExpressivE's capabilities to capture the core inference rules (Chapter 5), which are vital for representing the family relation properties of this use case.

Final Discussion. As we have seen, each of our three core contributions is vital to addressing the intricacies of this use case. On the one hand, to predict missing family links, this use case calls for KGEs that should be expressive enough (e.g., ExpressivE introduced in Chapter 4) to support vital inference rules of the family domain, including hierarchy and general composition while being efficient enough (e.g., SpeedE introduced in Chapter 5) to handle large KGs such as the one provided by the Central Bank of Italy. On the other hand, this use case demands the support of ontological querying over financial KGs with SPARQL and the power of Datalog-based inference, such as full recursion, to query and reason over close links. Thus, naturally, KG management solutions that combine the benefits of both the SW and DB world (e.g., SparqLog introduced in Chapter 6) are required. In total, this use case demonstrates how KG management systems and KGEs benefit from each other, (*i*) exploiting the ability to query and reason over highly complex scenarios via rule-based knowledge while (*ii*) enriching the factual, and possibly even ontological information, via embedding-based plausibility estimations.

1.7 Overall Goal and Organization

The aim of this dissertation is to bridge the reasoning, scalability, and data management divides between the KG research of the ML, DB, and SW communities. In particular, to address the problems raised by these gaps, we first introduce some formal background in Chapter 2. Building on these notions, we discuss the current state of the art in Chapter 3. Next, we address each of the divides in Chapters 4 to 6 and discuss how we achieved the corresponding research goals G1–G3. Finally, we summarize the key results of this dissertation and outline future work in Chapter 7.

CHAPTER 2

Background

As discussed in Chapter 1, the KG research of the Machine Learning (ML), Database (DB), and Semantic Web (SW) communities is divided along various dimensions. These divisions lead to different lenses that the communities use when looking at the representation and management of KGs, manifesting in the development of different technologies that suit the requirements visible through the respective lens. Each of the following sections discusses the views offered by the community-dependent lenses on the representation and management of KGs. We first consider the ML view in Section 2.1, then the SW view in Section 2.2, and finally, the DB view in Section 2.3. Note that the preliminaries discussed in Section 2.1 incorporate material from (Pavlović and Sallinger, 2023a,b, 2024a,b,c), and the preliminaries discussed in Sections 2.2 and 2.3 incorporate material from (Angles et al., 2023a,b). Furthermore, Sections 2.2 and 2.3 are partially taken from my Bachelor's (Pavlović, 2019) and Master's theses (Pavlović, 2020) and left unchanged here for consistency and readability.

2.1 ML View: Knowledge Graph Embeddings

This section introduces a typical view of the ML community on the representation of KGs. Specifically, it first introduces an ML-centered definition of KGs and follows by defining KGEs, the KGC problem, and evaluation methods (Abboud et al., 2020; Pavlović and Sallinger, 2023b, 2024c). In the context of the SW and DB communities, the following ML-centered KG definition roughly corresponds to the way that the (i) SW community sees RDF and (ii) DB community sees databases containing binary relations.

ML-centered KG Definition. Let us first introduce the *triple vocabulary* $\boldsymbol{T}$, consisting of a finite set of *entities* $\boldsymbol{E}$ and *relations* $\boldsymbol{R}$. We call an expression of the form $r_i(e_h, e_t)$ a triple, where $r_i \in \boldsymbol{R}$ and $e_h, e_t \in \boldsymbol{E}$. Furthermore, we call e_h the *head* entity and e_t the *tail* entity of the triple. Now, a KG G is a finite set of triples over $\boldsymbol{T}$, and KGC is the problem of predicting missing triples.

Knowledge Graph Embedding Models (KGEs) represent KGs as mathematical objects in latent vector spaces by assigning vectors (referred to as *embeddings*) to the entities and relations of a KG. KGEs define scoring functions $s\colon \boldsymbol{E} \times \boldsymbol{R} \times \boldsymbol{E} \to \mathbb{R}$ to quantify the plausibility of missing triples based on the learned embeddings. An *embedding instance* Θ represents a concrete set of parameter values instantiating a KGE's embeddings and, thus, its scoring function s_Θ. *Knowledge Graph Completion (KGC)* is the problem of predicting missing triples from the known triples of a KG. Although there are different flavors of KGC, the most common one is *link prediction* (Shen et al., 2022), which we shall define next. In particular, expressions of the form $r_i(e_h, ?)$ are called *tail queries*, and of the form $r_i(?, e_t)$ are called *head queries* with $r_i \in \boldsymbol{R}$ and $e_h, e_t \in \boldsymbol{E}$. Now, given a set of head and tail queries $\boldsymbol{Q}$, link prediction is the task of predicting the missing head or tail entity of each query in $\boldsymbol{Q}$. This corresponds approximately to answering queries over incomplete KGs, specifically atomic queries over binary relations in the DB world; and queries solely containing a single triple pattern in the SW world. For a complete correspondence to link prediction, the DB and SW queries need to be restricted to contain solely a single variable representing the possible head or tail entities. Following the literature (Balazevic et al., 2019b; Abboud et al., 2020; Charpenay and Schockaert, 2024), we will use the terms (i) link prediction and (ii) KGC interchangeably. KGEs have achieved promising performance on KGC and knowledge-driven applications (Wang et al., 2017; Broscheit et al., 2020).

Geometric Knowledge Graph Embedding Models (gKGEs) are special types of KGEs that represent entities and relations of a KG as geometric shapes in the semantic vector space, allowing for an intuitive *geometric interpretation* of their captured rules (Pavlović and Sallinger, 2023a,b, 2024a,b,c). Henceforth, we will focus on gKGEs as they typically offer more interpretability than other types of KGEs. Generally, KGEs can be evaluated by means of an (i) *experimental* evaluation on benchmark datasets, (ii) analysis of the model's *expressiveness*, and (iii) analysis of the *inference rules* that the model can capture. We will discuss each of these points next.

Experimental Evaluation. The experimental evaluation of KGEs requires a set of true and corrupted triples. True triples $r_i(e_h, e_t) \in G$ are corrupted by replacing either e_h or e_t with any $e_c \in \boldsymbol{E}$ such that the corrupted triple does not occur in G. KGEs define scores over triples and are optimized to score true triples higher than corrupted ones, thereby estimating a given triple's truth. More details on the standard benchmarks, evaluation protocol, and metrics for KGC are discussed in Section 2.1.1.

Expressiveness. A KGE is fully expressive if, for any finite set of disjoint true and false triples, a parameter set can be found such that the model classifies the triples of the set correctly. Intuitively, a fully expressive model can represent any given graph. However, this is not necessarily correlated with its inference capabilities (Abboud et al., 2020). For instance, while a fully expressive model may express the entire training set, it may have poor generalization capabilities (Abboud et al., 2020). Conversely, a model that is not fully expressive may underfit the training data severely (Abboud et al., 2020). Hence, KGEs should be both fully expressive and support important inference rules.

Inference Rules. The generalization capabilities of KGEs are commonly analyzed using inference rules (short: rules). An inference rule is a logical rule $\phi \Rightarrow \psi$, where ϕ is called its body and ψ its head. The following needs to hold for a rule $\phi \Rightarrow \psi$ to be satisfied over a graph G: if the rule's body is satisfied in G, then its head ψ must also be satisfied in G. Moreover, a rule of the form $\phi \Rightarrow \bot$ states that the rule ϕ is never satisfied in G. For instance, $r_1(X,Y) \wedge r_1(Y,X) \Rightarrow \bot$ represents that there is no pair of entities $X, Y \in \boldsymbol{E}$, such that both $r_1(X,Y) \in G$ and $r_1(Y,X) \in G$. Analyzing the rules that a KGE captures helps estimate its *inference capabilities* (Abboud et al., 2020). The captured inference rules of KGEs roughly correspond to the way in which (i) the SW community sees *axioms* and (ii) the DB community sees *dependencies* and *constraints*.

Intuition of Capturing. Following (Sun et al., 2019; Abboud et al., 2020; Pavlović and Sallinger, 2023b), a KGE *captures* an inference rule if there is an embedding instance such that the rule is captured (i) *exactly* and (ii) *exclusively*, as formalized for our KGEs in Sections 4.2.3 and 5.1.3. Capturing a rule means, at an intuitive level, that there is an embedding instance such that (i) if the instance satisfies the rule's body, then it also satisfies its head, and (ii) the instance does not capture any unwanted inference rule.

Core Inference Rules. Next, we briefly list logical rules that are vital for the DB and SW communities and commonly studied in the KGE literature (Sun et al., 2019; Abboud et al., 2020; Pavlović and Sallinger, 2023b, 2024c): (i) symmetry $r_1(X,Y) \Rightarrow r_1(Y,X)$, (ii) anti-symmetry $r_1(X,Y) \wedge r_1(Y,X) \Rightarrow \bot$, (iii) inversion $r_1(X,Y) \Leftrightarrow r_2(Y,X)$, (iv) general composition $r_1(X,Y) \wedge r_2(Y,Z) \Rightarrow r_3(X,Z)$, (v) hierarchy $r_1(X,Y) \Rightarrow r_2(X,Y)$, (vi) intersection $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y)$, and (vii) mutual exclusion $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow \bot$. Henceforth, we shall call these seven types of rules *core inference rules*.

2.1.1 KGC Benchmarks and Evaluation Protocol

This section briefly discusses benchmarks, the evaluation protocol, and metrics for KGC.

Traditional Benchmarks. Traditionally, KGC approaches were evaluated on the benchmarks FB15k and WN18 (Bordes et al., 2013), which contained many inverse relations, dramatically simplifying the KGC task (Toutanova and Chen, 2015). In particular, for a large portion of the test triples, their inverse triple would be directly contained in the training set, reducing the KGC task to learning to invert triples (Dettmers et al., 2018). To make the prediction task harder and more realistic, enhanced versions of these benchmarks — namely FB15k-237 (Toutanova and Chen, 2015) and WN18RR (Dettmers et al., 2018) — were derived by removing inverse relations. Following the literature (Sun et al., 2019; Abboud et al., 2020; Chami et al., 2020), we evaluated our models on these enhanced benchmark versions.

Rules in Standard Benchmarks. Several works examined the inference rules covered in these two enhanced benchmarks. The WN18RR benchmark retained most inference rules after its modification from WN18, including symmetry, anti-symmetry, hierarchy, and composition (Sun et al., 2019; Abboud et al., 2020; Song et al., 2021), making it an ideal candidate to evaluate whether KGEs can capture multiple inference rules jointly.

In contrast, FB15k-237 retained mostly composition rules (Sun et al., 2019; Abboud et al., 2020; Pavlović and Sallinger, 2023b), rendering it suitable to evaluate whether KGEs can capture (general) composition. Additionally, more recent works considered the YAGO3-10 (Mahdisoltani et al., 2015) benchmark, as it is the largest of the three datasets and, thus, the most interesting for evaluating the performance and scalability of KGEs (Abboud et al., 2020; Song et al., 2021). It contains a very challenging combination of almost all common inference rules (Abboud et al., 2020; Song et al., 2021).

Benchmark Domains. In terms of diversity, the domains of all three benchmarks are very different. Specifically, WN18RR is extracted from the WordNet database (Miller, 1995), representing lexical relations between English words, thus naturally containing many hierarchical relations (e.g., hypernym-of) (Chami et al., 2020). FB15k-237 is a subset of a collaborative database consisting of general knowledge (in English) called Freebase (Bollacker et al., 2007), which contains both hierarchical relations (e.g., part-of) and non-hierarchical ones (e.g., nationality) (Chami et al., 2020). YAGO3-10 is a subset of YAGO3, a KG describing people that, similarly to FB15k-237, contains both hierarchical relations (e.g., actedIn) and non-hierarchical relations (e.g., isMarriedTo).

Characteristics and Licenses. WN18RR, FB15k-237, and YAGO3-10 (Mahdisoltani et al., 2015) already provide a split into a training, validation, and testing set, which we directly adopted in any reported experiments. Table 2.1 lists some characteristics of these splits, precisely the number of training, validation, and testing triples. Furthermore, the table lists the number of entities and relations of each benchmark. Finally, concerning licensing, we did not find a license for WN18RR nor its superset WN18 (Bordes et al., 2013). Also, we did not find a license for FB15k-237, but we found that its superset FB15k (Bordes et al., 2013) uses the CC BY 2.5 license. For YAGO3-10, we also did not find a license, but we found that its superset, YAGO3 (Mahdisoltani et al., 2015), uses the CC BY 3.0 license.

Dataset	$\lvert\boldsymbol{E}\rvert$	$\lvert\boldsymbol{R}\rvert$	#training triples	#validation triples	#testing triples
FB15k-237	14,541	237	272,115	17,535	20,466
WN18RR	40,943	11	86,835	3,034	3,134
YAGO3-10	123,143	37	1,079,040	4,978	4,982

Table 2.1: Benchmark split characteristics: Number of entities, relations, and training, validation, and testing triples.

Evaluation Protocol and Metrics. In the standard KGC evaluation protocol as described by Sun et al. (2019); Balazevic et al. (2019b); Chami et al. (2020); Pavlović and Sallinger (2023b), the performance of KGEs is evaluated by measuring the ranking quality of each test set triple $r_i(e_h, e_t)$ over all possible heads e'_h and tails e'_t: $r_i(e'_h, e_t)$ for all $e'_h \in \boldsymbol{E}$ and $r_i(e_h, e'_t)$ for all $e'_t \in \boldsymbol{E}$. The typical metrics for measuring the KGC performance are the mean reciprocal rank (MRR) and H@k (Bordes et al., 2013). In particular, the filtered versions (Bordes et al., 2013) of these metrics are typically presented, i.e., all triples occurring in the training, validation, and testing set are deleted from the ranking (apart from the test triple that must be ranked), as scoring these triples highly, does not indicate a wrong inference. The most used metrics for assessing KGEs are the filtered MRR, H@1, H@3, and H@10 (Sun et al., 2019; Trouillon et al., 2016; Balazevic et al., 2019b; Abboud et al., 2020). Finally, we briefly review how these metrics are defined: The proportion of true triples among the predicted triples whose rank is at maximum k is represented by H@k, whereas the MRR reflects the average of inverse ranks ($1/rank$).

2.2 SW View: RDF and SPARQL

This section introduces a typical view of the SW community on KGs. Specifically, the SW community employs the following standard frameworks for managing KGs: (*i*) The Resource Description Framework (RDF) (Cyganiak et al., 2014) specifies a KG's graph component, representing links of a KG in the form of (subject, predicate, object) triples; (*ii*) the RDF Schema (RDFS) (Brickley and Guha., 2014) and Web Ontology Language (OWL) (Motik et al., 2012c,a) specify a KG's knowledge component, allowing to infer new triples from the ones defined in RDF, by employing vocabularies with explicit semantics; (*iii*) The SPARQL Protocol and RDF Query Language (SPARQL) (Prud'hommeaux and Seaborne, 2008; Harris and Seaborne, 2013) specifies a language for querying a KG's stored information. In the following, we briefly discuss notions of these frameworks that are relevant to this work. As the full formal syntax and model-theoretic semantics of these frameworks are not required here, we refer the interested reader for RDF to (Cyganiak et al., 2014; Hayes and Patel-Schneider, 2014), RDFS to (Brickley and Guha., 2014), OWL to (Motik et al., 2012c,b; Schneider, 2012), SPARQL to (Harris and Seaborne, 2013; Glimm and Ogbuji, 2013).

RDF (Cyganiak et al., 2014) is a W3C standard that defines a graph data model for describing Web resources. The RDF data model assumes three data domains: *Internationalized Resource Identifiers (IRIs))* that identify Web resources, *literals* that represent simple values, and *blank nodes* that identify anonymous resources. An *RDF triple* is a tuple (s, p, o), where s is the subject, p is the predicate, o is the object, all the components can be IRIs, the subject and the object can alternatively be a blank node, and the object can also be a literal. An *RDF graph* is a set of RDF triples. A *named graph* is an RDF graph identified by an IRI. An *RDF dataset* is a structure formed by a default graph and zero or more named graphs.

For example, consider that `<http://ex.org/film.rdf>` is an IRI that identifies an RDF graph with the RDF triples of Figure 2.1. This graph describes information about film crew members. Each line is an RDF triple, `<http://ex.org/glucas>` is an IRI, `"George"` is a literal, and `_:b1` is a blank node.

RDFS (Brickley and Guha., 2014) is a W3C standard, extending RDF with a vocabulary that adds explicit semantics to Web resources. For instance, RDFS allows the description of groups of related resources and predicates with triples that state the membership of a resource to a *class* and, respectively, of a predicate to a *property*. Formally, if `e`

```
<http://ex.org/glucas> <http://ex.org/name> "George"
<http://ex.org/glucas> <http://ex.org/lastname> "Lucas"
<http://ex.org/rwilliams> <http://ex.org/name> "Robin"
_:b1 <http://ex.org/name> "Robin"
```

Figure 2.1: Example of an RDF graph.

is a resource and `C` is a class, then `e rdf:type C` states that `e` is an instance of `C`. Analogously, if `p` is a predicate and `R` is a property, then `p rdf:type R` states that `p` is an instance of `R`. Furthermore, RDFS allows to state relationships between these classes and properties. For example, `A rdfs:subClassOf B` states that a class `A` is a subset of another class `B`. Analogously, `R rdfs:subPropertyOf S` states that a property `R` is a subset of another property `S`. Such relationships between classes and properties allow us to infer new information from a given RDF graph, as shown in the following example:

```
<http://ex.org/name> rdf:type rdf:Property
<http://ex.org/lastname> rdf:type rdf:Property
<http://ex.org/director> rdf:type rdfs:Class
<http://ex.org/crewMember> rdf:type rdfs:Class
<http://ex.org/director> rdfs:subClassOf <http://ex.org/crewMember>
<http://ex.org/actor> rdfs:subClassOf <http://ex.org/crewMember>
<http://ex.org/glucas> rdf:type <http://ex.org/director>
<http://ex.org/rwilliams> rdf:type <http://ex.org/actor>
```

`rdfs:Class` denotes the set of classes, analogously `rdf:Property` the set of properties. The example states that `<http://ex.org/name>` and `<http://ex.org/lastname>` are properties; and that `<http://ex.org/actor>`, `<http://ex.org/director>`, and `<http://ex.org/crewMember>` are classes, where the classes `<http://ex.org/actor>` and `<http://ex.org/director>` are subclasses of `<http://ex.org/crewMember>`. Since (*i*) `<http://ex.org/glucas>` is an instance of `<http://ex.org/director>` and (*ii*) `<http://ex.org/director>` is a subclass of `<http://ex.org/crewMember>`, RDFS allows us to infer that `<http://ex.org/glucas>` is also an instance of the class `<http://ex.org/crewMember>`. Analogously, the RDFS triples allow us to infer that `<http://ex.org/rwilliams>` is also an instance of `<http://ex.org/crewMember>`.

OWL (Motik et al., 2012a,c) is a W3C recommendation for defining ontologies. In particular, OWL 2 defines a more extensive vocabulary than RDFS, for instance, allowing the formalization of characteristics of properties, such as that a property is symmetric, asymmetric, inverse to another property, or disjoint with another property (cf., core inference rules of the ML view in Section 2.1). Moreover, there are different profiles of OWL 2 that essentially define subsets of OWL 2's features for efficiency reasons. On the one hand, OWL 2 Query Logic (OWL 2 QL) is a profile designed for efficient query answering. Specifically, it can answer queries in LOGSPACE with regard to the size of the KG's triples while providing many fundamental features for modeling characteristics of classes and properties, such as symmetric, asymmetric, inverse, and disjoint properties.

On the other hand, OWL 2 Rules Logic (OWL 2 RL) is a profile designed for scalable reasoning while aiming to keep most of OWL 2's expressive power.

SPARQL (Prud'hommeaux and Seaborne, 2008; Harris and Seaborne, 2013) is the W3C-recommended standard query language for RDF. The general structure of a SPARQL query is shown in Figure 2.2, where the `SELECT` clause defines the output of the query, the `FROM` clause defines the input of the query (i.e., an RDF dataset), and the `WHERE` clause defines a graph pattern.

```
SELECT ?N ?L
FROM <http://ex.org/film.rdf>
WHERE { ?X <http://ex.org/name> ?N .
        OPTIONAL { ?X <http://ex.org/lastname> ?L }}
ORDER BY ?N
```

Figure 2.2: Example of a SPARQL query.

The evaluation of a query begins with the construction of the RDF dataset to be queried, whose graphs are defined by one or more dataset clauses. A *dataset clause* is either an expression FROM u or FROM NAMED u, where u is an IRI that refers to an RDF graph. The former clause merges a graph into the default graph of the dataset, and the latter adds a named graph to the dataset.

The WHERE clause defines a graph pattern (GP). There are many types of GPs: triple patterns (RDF triples extended with variables), basic GPs (a set of GPs), optional GPs, alternative GPs (UNION), GPs on named graphs (GRAPH), negation of GPs (NOT EXISTS and MINUS), GPs with constraints (FILTER), existential GPs (EXISTS), and nesting of GPs (SubQueries). Moreover, a property path is a special GP that allows the expression of different types of reachability queries.

SPARQL employs *bag semantics*, which means that the results of evaluated SPARQL queries are not sets (set semantics) but *multisets*, preserving duplicate result entries. In particular, the result of evaluating a graph pattern is a multiset of solution mappings. A *solution mapping* is a set of variable-value assignments.

Next, we provide some intuition on the introduced concepts by evaluating the query depicted in Figure 2.2. First, the default graph and named graphs of the dataset are specified. Specifically, the query's FROM clause merges the triples of the RDF graph of Figure 2.1 into the dataset's empty default graph. As there are no GRAPH patterns in the WHERE clause, the specified GPs are executed over the default graph. The evaluation of the patterns over the default graph returns three mappings $\{\mu_1, \mu_2, \mu_3\}$, with $\mu_1(\texttt{?N}) = \texttt{"George"}$, $\mu_1(\texttt{?L}) = \texttt{"Lucas"}$, and $\mu_2(\texttt{?N}) = \mu_3(\texttt{?N}) = \texttt{"Robin"}$. Note that the duplicates μ_2 and μ_3 are preserved in the query result, as SPARQL employs bag semantics.

To clarify the difference between default and named graphs, consider the slightly modified query of Figure 2.3. Since the query contains solely a FROM NAMED clause (and no

FROM clause), the dataset's default graph is empty. Instead, the FROM NAMED clause adds the RDF graph of Figure 2.1 as the named graph `<http://ex.org/film.rdf>` to the dataset. Since the triple pattern `?X <http://ex.org/name> ?N.` is enclosed by a GRAPH pattern, the triple pattern is evaluated over the specified named graph, which is `<http://ex.org/film.rdf>`. Since the triple pattern within the OPTIONAL pattern is not enclosed in a GRAPH pattern, it is evaluated over the empty default graph. Thus, the evaluation of the query returns three mappings $\{\mu_1, \mu_2, \mu_3\}$, with $\mu_1(\texttt{?N}) = \texttt{"George"}$ and $\mu_2(\texttt{?N}) = \mu_3(\texttt{?N}) = \texttt{"Robin"}$. Note that in contrast to the result of Figure 2.2's query, the mapping $\mu_1(\texttt{?L}) = \texttt{"Lucas"}$ is missing in the solution multiset, as the triple pattern `?X <http://ex.org/lastname> ?L` is evaluated over the default graph, which is empty in this case.

```
SELECT ?N ?L
FROM NAMED <http://ex.org/film.rdf>
WHERE { GRAPH <http://ex.org/film.rdf> { ?X <http://ex.org/name> ?N. }
        OPTIONAL { ?X <http://ex.org/lastname> ?L }}
ORDER BY ?N
```

Figure 2.3: Example of a SPARQL query over a named graph.

The graph pattern matching step returns a multiset whose solution mappings are treated as a sequence without a specific order. Such a sequence can be arranged by using solution modifiers: ORDER BY allows to sort the solutions; DISTINCT eliminates duplicate solutions; OFFSET allows to skip a given number of solutions; and LIMIT restricts the number of output solutions.

Given the multiset of solution mappings, the final output is defined by a *query form*: SELECT projects the variables of the solutions; ASK returns *true* if the multiset of solutions is non-empty and *false* otherwise; CONSTRUCT returns an RDF graph whose content is determined by a set of triple templates; and DESCRIBE returns an RDF graph that describes the resources found.

Finally, observe that the semantics of SPARQL are defined based on matching subgraphs over plain RDF graphs. For ontological querying over KGs, using ontologies specified with RDFS and OWL, extensions of these *basic GP matching semantics* need to be found to not solely consider explicit subgraph structures but also entailment relations. To satisfy SPARQL's conditions on the evaluation results of basic GP matching, simplifications are necessary, limiting, for instance, the number of axiomatic triples that can contribute to a query solution. As the full formal details on standard entailment regimes for SPARQL are not required here, we refer the interested reader to (Glimm and Ogbuji, 2013).

2.3 DB View: Warded Datalog$^\pm$ and the Vadalog System

This section discusses a typical view of the DB community on KGs. In contrast to SW frameworks, the DB community employs different languages to manage KGs. In particular, languages such as Datalog and its extensions are used for representing, reasoning, and querying KGs. In the following, we discuss important notions of Datalog and some relevant dialects. For the definition of Datalog, we mostly follow the book of Abiteboul et al. (1995), adding some supplemental information from (Gottlob et al., 2012; Green et al., 2013; Gottlob and Pieris, 2015).

Datalog. First, let us assume disjoint sets of predicate and constant symbols over a domain *Dom* consisting of countably infinitely many constants. Furthermore, let us assume a mapping that maps each predicate symbol to its arity, which is always greater or equal to zero. Now, a term can be either a variable or a constant from *Dom*. Let P be a predicate symbol and let s be an n-tuple of terms; then, we call $P(s)$ an atom. The Datalog language allows rules of the form:

$$P(\bar{x}') \;\leftarrow\; P_1(\bar{x}_1), \ldots, P_n(\bar{x}_n),$$

where $\bar{x}' \subseteq \bigcup_i \bar{x}_i$ and where $P(\bar{x}')$ and $P_i(\bar{x}_i)$ with $1 \leq i \leq n$ are atoms. The atoms on the left of the arrow $(P_1(\bar{x}_1), \ldots, P_n(\bar{x}_n))$ are called the *body*, and the atom on the right $(P(\bar{x}'))$ is called the *head* of the rule. A finite set of Datalog rules is called a Datalog program Π. An expression without any variables is called a ground atom, ground rule, or ground program, respectively. A database D is a finite set of ground atoms.

Intuition. At an intuitive level, Datalog rules state how to infer new knowledge from the information encoded in databases. Let us consider a Datalog database D and rule $H \;\leftarrow\; B_1, \ldots, B_n$. Now, we can infer the rule's head atom H if all of the rule's body atoms $(B_1, \ldots, B_n)$ (i) occur in D or (ii) can be inferred in multiple steps from D via the rule. Analogously, given a Datalog database D and program Π, we can infer new atoms from D if all body atoms of a rule of Π (i) occur in D or (ii) can be inferred in multiple steps from D via rules from Π. Following common notation in first-order logic, we denote that an atom $P(\bar{x})$ can be inferred from D with Π as $\Pi \cup D \models P(\bar{x})$. For the full formal details, including operational, model-theoretic, and proof-theoretic semantics of Datalog, cf., e.g., (Abiteboul et al., 1995).

Example. To make these notions clear, let us consider the following example, consisting of a Datalog program Π_1 and a database D_1:

$$\begin{aligned}
\Pi_1 = \{&\mathit{SiblingOf}(X, Y) \leftarrow \mathit{BrotherOf}(X, Y)\\
&\mathit{SiblingOf}(X, Y) \leftarrow \mathit{SisterOf}(X, Y)\\
&\mathit{SiblingOf}(Y, X) \leftarrow \mathit{SiblingOf}(X, Y)\}\\
D_1 = \{&\mathit{SisterOf}(\mathit{ana}, \mathit{robin}), \mathit{BrotherOf}(\mathit{bob}, \mathit{alex})\}
\end{aligned}$$

The first two rules of the program Π_1 state that the *BrotherOf* and *SisterOf* relations are subsets of the *SiblingOf* relation, i.e., if X is a brother or sister of Y then X is also a sibling of Y. The third rule states that the *SiblingOf* relation is symmetric, i.e., if X is a sibling of Y, then Y is also a sibling of X. The database D_1 states that Ana is a sister of Robin and Bob is a brother of Alex. Now, from Π_1's second rule, we can infer $\mathit{SiblingOf}(ana, robin)$, i.e., that Ana is a sibling of Robin. From this inferred atom and Π_1's third rule, we can infer $\mathit{SiblingOf}(robin, ana)$. Analogously, we can infer that Bob is a sibling of Alex and vice versa.

Datalog Families. Calì et al. (2009) present Datalog$^\pm$ as a family of languages that extend Datalog (whence the +) to increase its expressive power but also impose restrictions (whence the −) to ensure decidability of answering Conjunctive Queries (CQs). The extension most relevant for our purposes is allowing *existential rules* of the form:

$$\exists \bar{z} P(\bar{x}', \bar{z}) \;\leftarrow\; P_1(\bar{x}_1), \ldots, P_n(\bar{x}_n),$$

with $\bar{x}' \subseteq \bigcup_i \bar{x}_i$, and $\bar{z} \cap \bigcup_i \bar{x}_i = \emptyset$. Allowing existential rules makes Datalog$^\pm$ well suited for capturing ontological reasoning.

Query Answering under Ontologies. Ontology-mediated query answering is defined by considering a given database D and program Π as logical theories. Query answering under an ontology defined by a Datalog$^\pm$ program comes down to solving an entailment problem. More precisely, let $Q(\vec{z})$ be a CQ with free variables $\vec{z}$ over database D and let an ontology be expressed by Datalog$^\pm$ program Π. Then the answers to $Q(\vec{z})$ over D under ontology Π are defined as $\{\vec{a} \mid \Pi \cup D \models Q(\vec{a})\}$, where $\vec{a}$ is a tuple of the same arity as $\vec{z}$ with values from D's domain.

Canonical Model and the Chase. Note that $\Pi \cup D$ can have many models. A canonical model is obtained via the *chase*, which is defined as follows: We say that a rule $\rho \in \Pi$ with head $p(\vec{z})$ is *applicable* to an instance I if there exists a homomorphism h from the body of ρ to I. We may then carry out a *chase step*, which consists of adding atom $h'(p(\vec{z}))$ to the instance I, where h' coincides with h on all variables occurring in the body of ρ and h' maps each existential variable in $p(\vec{z})$ to a fresh *labeled null* not occurring in I. A *chase sequence* for database D and program Π is a sequence of instances $I_0, I_1, \ldots$ obtained by applying a sequence of chase steps, starting with $I_0 = D$. The union of instances obtained by all possible chase sequences is referred to as $\mathit{Chase}(D, \Pi)$. The labeled nulls in $\mathit{Chase}(D, \Pi)$ play the same role as blank nodes in an RDF graph, i.e., resources for which the concrete value is not known. The importance of $\mathit{Chase}(D, \Pi)$ comes from the equivalence $\Pi \cup D \models Q(\vec{a}) \Leftrightarrow \mathit{Chase}(D, \Pi) \models Q(\vec{a})$ (Fagin et al., 2005). However, in general, $\mathit{Chase}(D, \Pi)$ is infinite. Hence, the previous equivalence does not yield an algorithm to evaluate a CQ $Q(\vec{z})$ w.r.t. database D and program Π. In fact, without restriction, this is an undecidable problem (Johnson and Klug, 1984). Several subclasses of Datalog$^\pm$ have, thus, been presented (Fagin et al., 2005; Calì et al., 2013, 2010a,b; Baget et al., 2009, 2011; Arenas et al., 2018) that ensure decidability of CQ answering (see (Calì et al., 2013) for an overview).

***Warded* Datalog$^\pm$.** One such subclass is *Warded* Datalog$^\pm$ (Arenas et al., 2018), which makes CQ answering even tractable (in terms of data complexity). For a formal definition of *Warded* Datalog$^\pm$, see (Arenas et al., 2018). We give the intuition of *Warded* Datalog$^\pm$

here. First, for all positions in rules of a program Π, distinguish if they are *affected* or not: a position is affected if the chase may introduce a labeled null here, i.e., a position in a head atom either with an existential variable or with a variable that occurs only in affected positions in the body. Then, for variables occurring in a rule ρ of Π, we identify the *dangerous* ones: a variable is dangerous in ρ if it may propagate a null in the chase, i.e., it appears in the head, and all its occurrences in the body of ρ are at affected positions. A Datalog$^{\pm}$ program Π is *warded* if all rules $\rho \in \Pi$ satisfy: either ρ contains no dangerous variable or all dangerous variables of ρ occur in a single body atom A (= the "ward") such that the variables shared by A and the remaining body occur in at least one non-affected position (i.e., they cannot propagate nulls).

Vadalog. Apart from the favorable computational properties, another critical aspect of Warded Datalog$^{\pm}$ is that a full-fledged engine (even with further extensions) exists: the Vadalog system (Bellomarini et al., 2018). It combines the full support of Warded Datalog$^{\pm}$ plus several extensions needed for practical use, including (decidable) arithmetics, aggregation, and other features. It has been deployed in numerous industrial scenarios in the finance, supply chain, and logistics sectors.

Bag Semantics of Datalog. Recall from Section 2.2 that SPARQL, the SW's standard querying language, employs bag semantics. However, Datalog-based languages typically employ set semantics (i.e., eliminate all duplicates in query results). In (Mumick et al., 1990), a bag semantics of Datalog was introduced based on *derivation trees.* Given a database D and Datalog program Π, a *derivation tree* (DT) is a tree T with node and edge labels, such that either (i) T consists of a single node labeled by an atom from D or (ii) Π contains a rule $\rho\colon\ H \leftarrow A_1, A_2, \ldots, A_k$ with $k > 0$, and there exist DTs $T_1, \ldots, T_k$ whose root nodes are labeled with atoms $C_1, \ldots, C_k$ such that $A_1, \ldots, A_k$ are simultaneously matched to $C_1, \ldots, C_k$ by applying some substitution θ, and T is obtained as follows: T has a new root node r with label $H\theta$ and the k root nodes of the DTs $T_1, \ldots, T_k$ are appended as child nodes of r in this order. All edges from r to its child nodes are labeled with ρ. Then, the bag semantics of program Π over database D consists of all ground atoms derivable from D by Π, and the multiplicity $m \in \mathbb{N} \cup \{\infty\}$ of each such atom A is the number of possible DTs with root label A. Datalog with bag semantics is readily extended by stratified negation (Mumick and Shmueli, 1993): the second condition of the definition of DTs now has to take negative body atoms in a rule $\rho\colon H \leftarrow A_1, A_2, \ldots, A_k, \neg B_1, \ldots \neg B_\ell$ with $k > 0$ and $\ell \geq 0$ with head atom H from some stratum i into account in that we request that none of the atoms $B_1\theta, \ldots B_\ell\theta$ can be derived from D via the rules in Π from strata less than i.

Bag Semantics via Set Semantics of Warded Datalog$^{\pm}$. In (Bertossi et al., 2019), it was shown how Datalog with *bag* semantics can be transformed into Warded Datalog$^{\pm}$ with *set* semantics. The idea is to replace every predicate $P(\ldots)$ with a new version $P(.\,; \ldots)$ with an extra, first argument to accommodate a labeled null, which is interpreted as tuple ID (TID). Each rule in Π of the form

$$\rho\colon H(\bar{x}) \ \leftarrow\ A_1(\bar{x}_1), A_2(\bar{x}_2), \ldots, A_k(\bar{x}_k), \text{ with } k > 0, \bar{x} \subseteq \cup_i \bar{x}_i$$

is then transcribed into the Datalog$^\pm$ rule

$$\rho' \colon \; \exists z \; H(z;\bar{x}) \leftarrow A_1(z_1;\bar{x}_1), A_2(z_2;\bar{x}_2), \ldots, A_k(z_k;\bar{x}_k),$$

with fresh, distinct variables $z, z_1, \ldots, z_k$. Some care (introducing auxiliary predicates) is required for rules with negated body atoms to solely produce rules with *safe negations*, i.e., rules where any variable in its head or in a negated atom must occur in at least one positive body atom of the same rule (Bertossi et al., 2019). A Datalog rule $\rho\colon H(\bar{x}) \leftarrow A_1(\bar{x}_1), \ldots, A_k(\bar{x}_k), \neg B_1(\bar{x}_{k+1}), \ldots, \neg B_\ell(\bar{x}_{k+\ell})$ with, $\bar{x}_{k+1}, \ldots, \bar{x}_{k+\ell} \subseteq \bigcup_{i=1}^{k} \bar{x}_i$ is replaced by $\ell + 1$ rules in the corresponding Datalog$^\pm$ program Π':

$$\begin{aligned}
\rho'_0 \colon \exists z H(z;\bar{x}) \leftarrow\; & A_1(z_1;\bar{x}_1), \ldots, A_k(z_k;\bar{x}_k), \\
& \neg Aux_1^\rho(\bar{x}_{k+1}), \ldots, \neg Aux_\ell^\rho(\bar{x}_{k+\ell}), \\
\rho'_i \colon Aux_i^\rho(\bar{x}_{k+i}) \leftarrow\; & B_i(z_i;\bar{x}_{k+i}), \quad i = 1, \ldots, \ell.
\end{aligned}$$

The resulting Datalog$^\pm$ program Π' is trivially warded since the rules thus produced contain no dangerous variables at all. Moreover, it is proved in (Bertossi et al., 2019) that an atom $P(\vec{a})$ is in the DT-defined bag semantics of Datalog program Π over database D with multiplicity $m \in \mathbb{N} \cup \{\infty\}$, iff $\mathit{Chase}(D, \Pi')$ contains atoms of the form $P(t;\vec{a})$ for m distinct labeled nulls t (i.e., the tuple IDs).

Discussion. Recall from Section 2.2 that the SW community represents properties of KGs using ontologies, e.g., defined in RDFS and OWL. Furthermore, recall that SPARQL, the SW's standard querying language for KGs, employs bag semantics, in contrast to Datalog-based languages of the DB community that typically employ set semantics. Warded Datalog$^\pm$'s and, thus, Vadalog's capabilities to (i) express bag semantics, (ii) answer queries under ontologies, while (iii) keeping CQ answering tractable; make the Vadalog system a promising base for developing a uniform and consistent KG management system for the DB and SW communities, as we shall see in Chapter 6.

CHAPTER 3

Related Work

In Chapter 1, we identified three dimensions that divide KG research from different communities. This section reviews related approaches that tried to bridge these gaps. As the following sections split the related work based on the dividing dimensions and their corresponding research goals, we recall them briefly in the following. Essentially, the reasoning divide states that while the DB and SW communities use a rich set of logical rules to express major data properties, contemporary KGEs from the ML community are strongly limited in their expressivity of such rules (cf., Table 1.1). This limitation calls for (G1) more expressive KGEs that can capture all core inference rules (defined in Section 2.1). Further problems arise from the scalability divide that points to the massive KGs the SW and DB communities provide, requiring (G2) the design of highly scalable and resource-efficient KGEs. On top of all this, the data management divide reveals a significant mismatch between SW and DB frameworks for modeling, querying, and reasoning over KGs. This technological mismatch makes blending KGs from both communities a complex endeavor, calling for (G3) a uniform and consistent KG management system.

The following sections discuss related work split by our research goals G1–G3. In particular, as G1 and G2 are concerned with proposing more expressive and efficient KGEs, Section 3.1 classifies KGEs into different families and reviews their expressivity, and Section 3.2 reviews the efficiency problems of KGEs. Next, Section 3.3 reviews approaches that took a step toward G3 by proposing either theoretical translations between standard SW and DB frameworks or developing practical systems that partially allow to query (and reason over) KGs from the DB and SW communities. Note that the related approaches studied in Sections 3.1 and 3.2 incorporate material from (Pavlović and Sallinger, 2023a,b, 2024a,b,c) and that the approaches analyzed in Sections 3.3 incorporate material from (Angles et al., 2023a,b). Furthermore, note that Section 3.3's content is partially taken from my Bachelor's (Pavlović, 2019) and Master's theses (Pavlović, 2020) and left unchanged here for consistency and readability.

3.1 G1: Expressive KGEs

The main focus of our work lies on geometric Knowledge Graph Embedding Models (gKGEs), i.e., KGEs that allow for a geometric interpretation of their captured inference rules. Thus, we have excluded neural KGEs as they are typically less interpretable (Dettmers et al., 2018; Socher et al., 2013; Nathani et al., 2019; Wang et al., 2021). gKGEs are commonly classified by how they embed relations:

Functional Models. So far, solely a subset of translational models supports composition. We call this subset *functional models*, as they embed relations as functions $\boldsymbol{f_{r_i}} : \mathbb{K}^d \rightarrow \mathbb{K}^d$ and entities as vectors $\boldsymbol{e_j} \in \mathbb{K}^d$ over some field $\mathbb{K}$. These models represent true triples $r_i(e_h, e_t)$ as $\boldsymbol{e_t} = \boldsymbol{f_{r_i}}(\boldsymbol{e_h})$. Thereby, they can capture composition rules via functional composition. TransE (Bordes et al., 2013) is the pioneering functional model, embedding relations r_i as $\boldsymbol{f_{r_i}}(\boldsymbol{e_h}) = \boldsymbol{e_h} + \boldsymbol{e_{r_i}}$ with $\boldsymbol{e_{r_i}} \in \mathbb{K}^d$. However, it is neither fully expressive nor can it capture *1–N*, *N–1*, *N–N*, nor symmetric relations. RotatE (Sun et al., 2019) embeds relations as rotations in complex space, allowing it to capture symmetry rules but leaving it otherwise with TransE's limitations. Recently, it was discovered that TransE and RotatE may only capture a fairly limited notion of composition (Zhang et al., 2019; Abboud et al., 2020; Lu and Hu, 2020; Gao et al., 2020), cf. also Section 4.3.1. Therefore, extensions have been proposed to tackle some limitations, such as MuRP (Balazevic et al., 2019a), RotH (Chami et al., 2020), HAKE (Zhang et al., 2020), and ConE (Bai et al., 2021). While these extensions solved some limitations, the purely functional nature of TransE, RotatE, and any of their extensions limits them to capture solely *compositional definition* and not *general composition* (see Table 1.1 for the defining formulas and cf. also Section 4.3.1 for details). Therefore, capturing general composition is still an open problem. Even more, functional models are incapable of capturing vital rules, such as hierarchies, completely (Abboud et al., 2020).

Bilinear Models embed relations as matrices, allowing them to factorize a graph's adjacency matrix by computing the bilinear product of entity and relation embeddings. The pioneering bilinear model is RESCAL (Nickel et al., 2011). It embeds relations with full-rank $d \times d$ matrices $\boldsymbol{M}$ and entities with d-dimensional vectors. However, its parameter size grows quadratically with its dimensionality d, limiting RESCAL's scalability (Kazemi and Poole, 2018). Thus, more scalable bilinear gKGEs were proposed, such as DistMult (Yang et al., 2015a), HolE (Nickel et al., 2016), ComplEx (Trouillon et al., 2016), TuckER (Balazevic et al., 2019b), SimplE (Kazemi and Poole, 2018), QuatE (Zhang et al., 2019), and DualQuatE (Cao et al., 2021). In particular, DistMult (Yang et al., 2015a) constrains RESCAL's relation matrix $\boldsymbol{M}$ to a diagonal matrix for efficiency reasons, limiting DistMult to capture symmetric relations only. HolE (Nickel et al., 2016) solves this limitation by combining entity embeddings via circular correlation, whereas ComplEx (Trouillon et al., 2016) solves this limitation by embedding relations with a complex-valued diagonal matrix. HolE and ComplEx have subsequently been shown to be equivalent (Hayashi and Shimbo, 2017). SimplE (Kazemi and Poole, 2018) is based on canonical polyadic decomposition (Hitchcock, 1927). TuckER (Balazevic et al., 2019b) is based on Tucker decomposition (Tucker, 1966) and extends the capabilities of RESCAL and SimplE (Balazevic et al., 2019b). QuatE (Zhang et al., 2019), and DualQuatE (Cao et al., 2021) extend ComplEx by using extensions of the complex space as embedding spaces. In particular, QuatE uses the quaternion space, which extends the real space by three imaginary units, whereas DualQuatE uses the dual-quaternion space, which extends the real space by seven imaginary units. While all bilinear models, excluding DistMult, are fully expressive, they cannot capture any notion of composition.

Spatial Models. Spatial models define semantic regions within the embedding space that allow the intuitive representation of certain rules. In entity classification, for example, bounded axis-aligned hyper-rectangles (boxes) represent entity classes, capturing class hierarchies through the spatial subsumption of these boxes (Vilnis et al., 2018; Subramanian and Chakrabarti, 2018; Li et al., 2019). Also, query answering systems — such as Query2Box (Ren et al., 2020) — have used boxes to represent answer sets due to their natural interpretation as sets of entities. Although Query2Box can be used for KGC, entity classification approaches cannot scalably be employed in the general KGC setting, as this would require an embedding for each entity tuple (Abboud et al., 2020). BoxE (Abboud et al., 2020) is the first spatial KGE dedicated to KGC. It embeds relations as a pair of boxes and entities as a set of points and bumps in the embedding space. The usage of boxes enables BoxE to capture most core inference rules, specifically those that can be described by the intersection of boxes in the embedding space, such as hierarchy. Moreover, boxes enable BoxE to capture *1–N*, *N–1*, and *N–N* relations naturally. Yet, BoxE cannot capture any notion of composition (Abboud et al., 2020).

Our Work. Recently, we proposed ExpressivE (Pavlović and Sallinger, 2023b), a *spatio-functional gKGE* that combines the advantages of both spatial and functional models by embedding relations as hyper-parallelograms. In Chapter 4, we show that ExpressivE captures all core inference rules simultaneously while additionally allowing to display any captured inference rule — *including general composition* — through the spatial relation of hyper-parallelograms. Thus, Chapter 4 shows how we reach G1.

3.2 G2: Efficient KGEs

Although the KGE families discussed in Section 3.1 are vastly different, all contemporary gKGEs suffer from at least one of two efficiency problems, namely the *embedding space* and *high-dimensionality problem*. The following paragraphs discuss each of them in detail.

Embedding Space Problem. Contemporary gKGEs often overcome the limitations of former ones by exploring increasingly *more complex* spaces. For example, while (*i*) RESCAL and DistMult use the Euclidean space $\mathbb{R}$, (*ii*) ComplEx uses the complex space, extending $\mathbb{R}$ by one imaginary unit, (*iii*) QuatE uses the quaternion space, extending $\mathbb{R}$ by three imaginary units, and (*iv*) DualQuatE uses the dual-quaternion space, extending $\mathbb{R}$ by seven imaginary units. Thus, a d-dimensional entity embedding of (*i*) RESCAL and DISTMULT requires d, (*ii*) ComplEx requires $2d$, (*iii*) QuatE requires $4d$, and (*iv*) DualQuatE requires even $8d$ real-valued parameters. Therefore, gKGEs based in more complex embedding spaces typically require more parameters, lowering their efficiency compared to Euclidean gKGEs (Wang et al., 2021).

High-Dimensionality Problem. Even more, most gKGEs require *high-dimensional* embeddings to reach good KGC performance (Chami et al., 2020; Wang et al., 2021). Yet, high embedding dimensionalities of 200, 500, or 1000 (Sun et al., 2019; Zhang et al., 2019) increase the time and space requirements of gKGEs drastically, limiting their efficiency and application to resource-constrained environments, especially in mobile smart devices (Wang et al., 2021).

Hyperbolic gKGEs such as RotH and AttH (Chami et al., 2020) embed entities and relations in the hyperbolic space, which allows for high-fidelity and parsimonious representations of *hierarchical relations* (Balazevic et al., 2019a; Chami et al., 2020), i.e., relations that describe hierarchies between entities, such as *part_of*. This allowed them to reach promising KGC performance using low-dimensional embeddings, addressing the high-dimensionality problem (Chami et al., 2020). Yet, most hyperbolic gKGEs were limited to expressing a single global entity hierarchy per relation. ConE (Bai et al., 2021) solves this problem by embedding entities as hyperbolic cones and relations as transformations between these cones. However, hyperbolic gKGEs typically cannot directly employ Euclidean addition and scalar multiplication but require far more costly hyperbolic versions of these operations, termed Möbius Addition and Multiplication. Thus, they fail to address the embedding space problem, which results in high time requirements for hyperbolic gKGEs (Wang et al., 2021).

Euclidean gKGEs have recently shown strong representation, inference, and KGC capabilities under high-dimensional conditions. On the one hand, HAKE (Zhang et al., 2020) achieved promising results for representing hierarchical relations on which hyperbolic gKGEs are typically most effective. On the other hand, our recent ExpressivE model (Pavlović and Sallinger, 2023b) managed to capture all core inference rules (see Chapter 4 for details). Although Euclidean gKGEs address the embedding space problem, their reported KGC results under low dimensionalities are dramatically lower than those of hyperbolic gKGEs (Chami et al., 2020). Thus, they currently fail to address the high-dimensionality problem.

Our Work. Based on the discussion above, we find that no contemporary gKGE addresses both the embedding space and high dimensionality problems simultaneously. Inspired by ExpressivE's promising results with high-dimensional embeddings (cf., (Pavlović and Sallinger, 2023b) and see Chapter 4), we propose SpeedE, a highly resource-efficient gKGE that *jointly* focuses on both efficiency problems, thereby reaching G2. Chapter 5 presents the SpeedE model and how we reach G2 in detail.

3.3 G3: Uniform Knowledge Graph Management

This section reviews approaches that took a step toward G3, i.e., developing a uniform and consistent KG management system. Recall the SW and DB communities' requirements

(R1–R5) for such a uniform system, identified in Chapter 1. For the reader's convenience, we briefly restate the five identified requirements here. To be accepted by the SW and DB communities, a uniform KG management system should support (R1) the most common SPARQL features for querying KGs; (R2) bag semantics to answer queries according to the SPARQL standard (Prud'hommeaux and Seaborne, 2008; Harris and Seaborne, 2013); (R3) OWL 2 QL for ontological reasoning, which technically means for DB languages such as Datalog that existential quantification in rule heads is required; and (R4) full recursion to provide the expressive power necessary for complex business cases. Finally, (R5) an implemented system is required to apply the framework in the real world. In the following sections, we review existing approaches and analyze which of the five requirements they satisfy and which they fall short of. In particular, Section 3.3.1 reviews theoretical endeavors targeted toward G3, while Section 3.3.2 reviews practical systems that attempted to combine SW and DB frameworks for managing KGs.

3.3.1 Theoretical Approaches

Several theoretical research efforts have aimed at G3, i.e., bridging the gap between the DB and SW communities for managing KGs.

Translations of SPARQL to Answer Set Programming. In a series of papers, Polleres et al. presented translations of SPARQL and SPARQL 1.1 to various extensions of Datalog. The first translation from SPARQL to Datalog (Polleres, 2007) converted SPARQL queries into Datalog programs by employing negation as failure. This translation was later extended by the addition of new features of SPARQL 1.1 and by considering its bag semantics in (Polleres and Wallner, 2013). Thereby, Polleres and Wallner created a nearly complete translation of SPARQL 1.1 queries to Datalog with Disjunction (DLV) programs. However, the translation had two major drawbacks: On the one hand, the chosen target language, DLV, does not support ontological reasoning as it does not contain existential quantification, thereby missing (R3) a key requirement of the Semantic Web community. On the other hand, (R5) the requirement of an implemented system is only partially fulfilled since the prototype implementation *DLVhex-SPARQL Plugin* (Polleres and Schindlauer, 2007) of the SPARQL to Datalog translation of (Polleres, 2007) has not been extended to cover SPARQL 1.1's new features.

Alternative Translations of SPARQL to Datalog. An alternative approach of relating SPARQL to non-recursive Datalog with stratified negation (or, equivalently, to Relational Algebra) was presented by Angles and Gutierrez in (Angles and Gutierrez, 2008). The peculiarities of negation in SPARQL were treated in a separate paper (Angles and Gutierrez, 2016b). The authors later extended this line of research to an exploration of the bag semantics of SPARQL and a characterization of the structure of its algebra and logic in (Angles and Gutierrez, 2016a). They translated a few SPARQL features into a Datalog dialect with bag semantics (multiset non-recursive Datalog with safe negation). This work considered only a small set of SPARQL functionality on a very abstract level and used again a target language that does not support ontological reasoning, failing to meet important requirements (R1, R3) of the SW community. Most importantly, no implementation of the translations provided by Angles and Gutierrez exists, thus failing to fulfill R5.

Supporting Ontological Reasoning via Existential Rules. In (Calì et al., 2009), Datalog$^\pm$ was presented as a family of languages that are particularly well suited for capturing ontological reasoning. The "+" in Datalog$^\pm$ refers to the crucial extension compared with Datalog by *existential rules*, that is, allowing existentially quantified variables in the rule heads. However, without restrictions, basic reasoning tasks such as answering Conjunctive Queries w.r.t. an ontology given by a set of existential rules become undecidable (Johnson and Klug, 1984). Hence, numerous restrictions have been proposed (Fagin et al., 2005; Calì et al., 2013, 2010a,b; Baget et al., 2009, 2011) to ensure the decidability of such tasks, which led to the "−" in Datalog$^\pm$. Of all variants of Datalog$^\pm$, Warded Datalog$^\pm$ (Arenas et al., 2018) ultimately turned out to constitute the best compromise between complexity and expressiveness, and it has been implemented in an industrial-strength system – the Vadalog system (Bellomarini et al., 2018), thus fulfilling requirement R5. However, the requirements of (R1) supporting SPARQL with or without (R2) bag semantics have not been fulfilled up to now.

Warded Datalog$^\pm$ with Bag Semantics. In (Bertossi et al., 2019), it was shown that Warded Datalog$^\pm$ using set semantics can be used to represent Datalog using bag semantics by using existential quantification to introduce new tuple IDs. It was assumed that these results could be leveraged for future translations from SPARQL with bag semantics to Warded Datalog$^\pm$ with set semantics. However, the (R1) theoretical translation of SPARQL to Vadalog using these results and also (R5) its implementation by extending Vadalog were left open in (Bertossi et al., 2019) and considered of primary importance for future work.

3.3.2 Practical Approaches

Several systems have aimed at bridging the gap between DB and SW technologies. The World Wide Web Consortium (W3C) lists *StrixDB*, *DLVhex SPARQL-engine*, and *RDFox* as systems that support SPARQL in combination with Datalog[1]. Furthermore, we also have a look at ontological reasoning systems *Vadalog*, *Graal*, and *VLog*, which either understand SPARQL to some extent or, at least in principle, could be extended in order to do so.

DLVhex-SPARQL Plugin. The DLVhex-SPARQL Plugin (Polleres and Schindlauer, 2007) is a prototype implementation of the SPARQL to Datalog translation in (Polleres, 2007). According to the repository's ReadMe file[2], it supports basic graph patterns, simple conjunctive *FILTER* expressions (such as *ISBOUND*, *ISBLANK*, and arithmetic comparisons), the *UNION*, *OPTIONAL*, and *JOIN* operation. Other operations, language tags, etc. are not supported, and query results do not conform to the SPARQL protocol,

[1]https://www.w3.org/wiki/SparqlImplementations (last visited 09/25/2023)
[2]https://sourceforge.net/p/dlvhex-semweb/code/HEAD/tree/dlvhex-sparqlplugin/trunk/README (last visited 09/25/2023)

according to the ReadMe file. Thus, the DLVhex-SPARQL Plugin misses many vital (R1) SPARQL features. Moreover, the limited support of existential quantification (see (Eiter et al., 2013)) by DLV does not suffice for (R3) ontological reasoning as required by the OWL 2 QL standard.

RDFox. RDFox is an RDF store developed and maintained at the University of Oxford (Nenov et al., 2015). It reasons over OWL 2 RL ontologies in Datalog and computes/stores materializations of the inferred consequences for efficient query answering (Nenov et al., 2015). The answering process of SPARQL queries is not explained in great detail, except that queries are evaluated on top of these materializations by employing different scanning algorithms (Nenov et al., 2015). However, translating SPARQL to Datalog – for representing, querying, and reasoning over KGs in one consistent system – is not supported[3]. Moreover, RDFox does currently not support (R1) property paths and some other SPARQL 1.1 features[4].

StrixDB. StrixDB is an RDF store developed as a simple tool for working with middle-sized RDF graphs, supporting SPARQL 1.0 and Datalog reasoning capabilities[5]. To the best of our knowledge, no academic paper or technical report explains the system's capabilities in greater detail. The *StrixStore* documentation page[6] lists examples of how to integrate Datalog rules into SPARQL queries and query graphs enhanced by Datalog ontologies. However, translating SPARQL to Datalog is not supported[7]. Moreover, important (R1) SPARQL 1.1 features such as aggregation and property paths are not supported by StrixDB.

Graal. Graal was developed as a toolkit for querying ontologies with existential rules (Baget et al., 2015). The system does not focus on a specific storage system but specializes in algorithms that can answer queries regardless of the underlying database type (Baget et al., 2015). It reaches this flexibility by translating queries from their host system language into Datalog$^{\pm}$. However, it pays the trade-off of restricting itself to answering conjunctive queries only (Baget et al., 2015) and therefore supports merely a small subset of SPARQL features[8] — e.g., (R1) basic features such as *UNION* or *MINUS* are missing.

VLog. VLog is a rule engine developed at the TU Dresden (Carral et al., 2019). The system transfers incoming SPARQL queries to specified external SPARQL endpoints such as Wikidata and DBpedia and incorporates the received query results into their knowledge base (Carral et al., 2019). Therefore, the responsibility of query answering is handed over to RDF triple stores that provide a SPARQL query answering endpoint, thus failing to provide (R5) a uniform, integrated framework for combining query answering with ontological reasoning.

[3]see https://docs.oxfordsemantic.tech/reasoning.html (last visited 09/25/2023)
[4]https://docs.oxfordsemantic.tech/3.1/querying-rdfox.html#query-language (last visited 09/25/2023)
[5]http://opoirel.free.fr/strixDB/ (last visited 09/25/2023)
[6]http://opoirel.free.fr/strixDB/DOC/StrixStore_doc.html (last visited 09/25/2023)
[7]see http://opoirel.free.fr/strixDB/dbfeatures.html (last visited 09/25/2023)
[8]https://graphik-team.github.io/graal/ (last visited 09/25/2023)

The Vadalog system (Bellomarini et al., 2018) is a KG management system implementing the logic-based language Warded Datalog$^{\pm}$. It extends Datalog by including existential quantification necessary for ontological reasoning while maintaining reasonable complexity. As an extension of Datalog, it supports full recursion. Although Warded Datalog$^{\pm}$ has the capabilities to support SPARQL 1.1 under the OWL 2 QL entailment regime (Arenas et al., 2018) (*considering set semantics*), no complete theoretical nor any practical translation from SPARQL 1.1 to Warded Datalog$^{\pm}$ exists. Therefore, the (R2) bag semantics and (R1) SPARQL feature coverage requirements are not met.

Our Work. As we have seen in the previous sections, no uniform and consistent framework for managing KGs exists. Inspired by Vadalog's theoretical capabilities to support SPARQL 1.1 under OWL 2 QL in set semantics, we design the SparqLog system. It translates RDF graphs and SPARQL 1.1 queries to Vadalog programs. As Vadalog additionally allows the representation of RDFS/OWL ontologies as programs, SparqLog brings representing, querying, and reasoning over KGs together in one consistent system. As Chapter 6 shows, SparqLog satisfies all identified requirements (R1–R5), rendering it a uniform system for managing KGs, thereby reaching G3.

CHAPTER 4

Reasoning Divide

This chapter presents how we overcome the reasoning divide, reaching (G1) the goal of designing a KGE that captures logical rules relevant to the DB and SW community, specifically the core inference rules (as discussed in Section 1.1). In particular:

- We introduce the *spatio-functional* embedding model **ExpressivE**. It embeds pairs of entities as points and relations as *hyper-parallelograms* in the space $\mathbb{R}^{2d}$, which we call the *virtual triple space*. The virtual triple space allows ExpressivE to represent rules through the spatial relationship of hyper-parallelograms, offering an intuitive and consistent geometric interpretation of ExpressivE embeddings and their captured rules.
- To reach G1, we prove that ExpressivE can capture all core inference rules. This makes ExpressivE the first model capable of capturing both general composition and hierarchy jointly.
- Additionally, we prove that our model is **fully expressive**, making ExpressivE the first KGE that both supports composition and is fully expressive.
- Finally, we evaluate ExpressivE on the two standard KGC benchmarks WN18RR (Dettmers et al., 2018) and FB15k-237 (Toutanova and Chen, 2015), revealing that ExpressivE is competitive with state-of-the-art gKGEs and even significantly outperforms them on WN18RR.

Organization. This chapter contains material from the following constituent papers of this dissertation (Pavlović and Sallinger, 2023a,b, 2024a,b,c). Based on this material, Section 4.1 introduces ExpressivE, the virtual triple space, and interprets our model's parameters within it. Section 4.2 analyzes our model's expressive power and inference capabilities. Section 4.3 discusses ExpressivE's two natures and how their combination allows ExpressivE to capture general composition and hierarchy jointly. Section 4.4 presents the experimental setup of the subsequent analyzes. Section 4.5 discusses empirical results together with our model's space complexity, and Section 4.6 summarizes our work.

4.1 ExpressivE and the Virtual Triple Space

This section introduces ExpressivE, a gKGE targeted toward KGC with the capabilities of capturing a rich set of inference rules. ExpressivE embeds entities as *points* and relations as hyper-parallelograms in the virtual triple space $\mathbb{R}^{2d}$. More concretely, instead of analyzing our model in the d-dimensional embedding space $\mathbb{R}^d$, we construct the novel *virtual triple space* that grants ExpressivE's parameters a geometric meaning. Above all, the virtual triple space allows us to intuitively interpret ExpressivE embeddings and their captured rules, as discussed in Section 4.2.

Representation. Entities $e_j \in \boldsymbol{E}$ are embedded in ExpressivE via a vector $\boldsymbol{e_j} \in \mathbb{R}^d$, representing points in the latent embedding space $\mathbb{R}^d$. Relations $r_i \in \boldsymbol{R}$ are embedded as hyper-parallelograms in the virtual triple space $\mathbb{R}^{2d}$. More specifically, ExpressivE assigns to a relation r_i for each of its arity positions $p \in \{h, t\}$ the following vectors: (1) a *slope vector* $\boldsymbol{s_i^p} \in \mathbb{R}^d$, (2) a *center vector* $\boldsymbol{c_i^p} \in \mathbb{R}^d$, and (3) a *width vector* $\boldsymbol{w_i^p} \in (\mathbb{R}_{\geq 0})^d$. Intuitively, these vectors define the slopes $\boldsymbol{s_i^p}$ of the hyper-parallelogram's boundaries, its center $\boldsymbol{c_i^p}$ and width $\boldsymbol{w_i^p}$. A triple $r_i(e_h, e_t)$ is captured to be true in an ExpressivE model if its relation and entity embeddings satisfy the following inequalities:

$$(\boldsymbol{e_h} - \boldsymbol{c_i^h} - \boldsymbol{s_i^t} \odot \boldsymbol{e_t})^{|.|} \preceq \boldsymbol{w_i^h} \tag{4.1}$$

$$(\boldsymbol{e_t} - \boldsymbol{c_i^t} - \boldsymbol{s_i^h} \odot \boldsymbol{e_h})^{|.|} \preceq \boldsymbol{w_i^t} \tag{4.2}$$

Where $\boldsymbol{x}^{|.|}$ represents the element-wise absolute value of a vector $\boldsymbol{x}$, $\odot$ represents the Hadamard (i.e., element-wise) product, and $\preceq$ represents the element-wise less or equal operator. It is very complex to interpret this model in the embedding space $\mathbb{R}^d$. Hence, we construct followingly a *virtual triple space* in $\mathbb{R}^{2d}$ that will ease reasoning about the parameters and inference capabilities of ExpressivE.

Virtual Triple Space. We construct this virtual space by concatenating the head and tail entity embeddings. In detail, this means that any pair of entities $(e_h, e_t) \in \boldsymbol{E} \times \boldsymbol{E}$ defines a point in the virtual triple space by concatenating their entity embeddings $\boldsymbol{e_h}, \boldsymbol{e_t} \in \mathbb{R}^d$, i.e., $(\boldsymbol{e_h} || \boldsymbol{e_t}) \in \mathbb{R}^{2d}$, where $||$ is the concatenation operator. We will henceforth call the first d dimensions of the virtual triple space the *head dimensions* and the second d dimensions the *tail dimensions*. A set of important sub-spaces of the virtual triple space are the 2-dimensional spaces, created from the j-th embedding dimension of head entities and the j-th dimension of tail entities — i.e., the j-th and $(d+j)$-th virtual triple space dimensions. We call them *correlation subspaces*, as they visualize the captured relation-specific dependencies of head and tail entity embeddings, as will be discussed followingly. Moreover, we call the correlation subspace spanned by the j-th and $(d+j)$-th virtual triple space dimension the j-th correlation subspace.

Parameter Interpretation. Inequalities 4.1 and 4.2 construct each an intersection of two parallel half-spaces in any correlation subspace of the virtual triple space. We call the intersection of two parallel half-spaces a *band*, as they are limited by two parallel boundaries. Henceforth, we will denote with $\boldsymbol{v}(j)$ the j-th dimension of a vector $\boldsymbol{v}$. For example, $(\boldsymbol{e_h}(j) - \boldsymbol{c_i^h}(j) - \boldsymbol{s_i^t}(j) \odot \boldsymbol{e_t}(j))^{|.|} \preceq \boldsymbol{w_i^h}(j)$ defines a band in the j-th correlation subspace. The intersection of two bands results either in a band (if one band subsumes the other) or a parallelogram. Since we are interested in constructing ExpressivE embeddings that capture certain inference rules, it is sufficient to consider parallelograms for these constructions. Figure 4.1a visualizes a relation parallelogram (green solid) and its parameters (orange dashed) in the j-th correlation subspace. In essence, the parallelogram is the result of the intersection of two bands (thick blue and magenta lines), where its boundaries' slopes are defined by $\boldsymbol{s_i^p}$, the center of the parallelogram is defined by $\boldsymbol{c_i^p}$, and finally, the widths of each band are defined by $\boldsymbol{w_i^p}$.

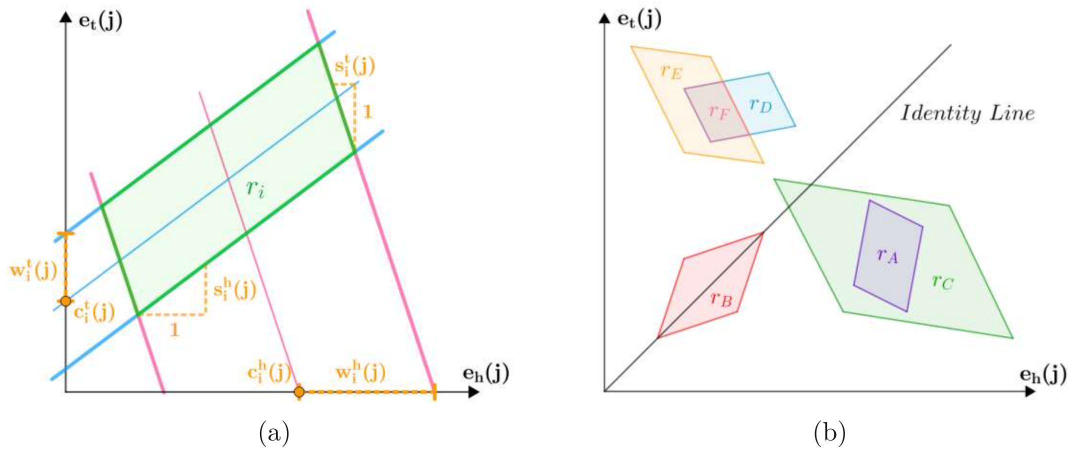

Figure 4.1: (a) Interpretation of relation parameters (orange dashed) as a parallelogram (green solid) in the j-th correlation subspace; (b) Multiple relation embeddings with the following properties: Symmetry (r_B), Anti-Symmetry (r_A, r_D, r_E, r_F), Inversion ($r_D(X,Y) \Leftrightarrow r_A(Y,X)$), Hierarchy ($r_A(X,Y) \Rightarrow r_C(X,Y)$), Intersection ($r_D(X,Y) \wedge r_E(X,Y) \Rightarrow r_F(X,Y)$), Mutual Exclusion (e.g., $r_A(X,Y) \wedge r_B(X,Y) \Rightarrow \perp$).

Since Inequalities 4.1 and 4.2 solely capture dependencies within the same dimension, any two different dimensions $j \neq k$ of head and tail entity embeddings are independent. Thus, relations are embedded as hyper-parallelograms in the virtual triple space, whose edges are solely crooked in any j-th correlation subspace. Intuitively, the crooked edges represent relation-specific dependencies between head and tail entities and are thus vital for the expressive power of ExpressivE. Note that each correlation subspace represents one dimension of the element-wise Inequalities 4.1 and 4.2. Since the sum of all correlation subspaces represents all dimensions of Inequalities 4.1 and 4.2, it is sufficient to analyze all correlation subspaces to identify the captured inference rules of an ExpressivE model.

Distance Function. ExpressivE employs the typical distance function of spatial gKGEs $D : \boldsymbol{E} \times \boldsymbol{R} \times \boldsymbol{E} \to \mathbb{R}^{2d}$ (Abboud et al., 2020) — measuring the distance of entity pair embeddings (points) to relation embeddings (hyper-parallelograms) —– which we define next. Let $\boldsymbol{\tau_{r_i(h,t)}}$ denote the embedding of a triple $r_i(h,t)$, i.e., $\boldsymbol{\tau_{r_i(h,t)}} = (\boldsymbol{e_{ht}} - \boldsymbol{c_i} - \boldsymbol{s_i} \odot \boldsymbol{e_{th}})^{|.|}$, with $\boldsymbol{e_{ht}} = (\boldsymbol{e_h}||\boldsymbol{e_t})$, $\boldsymbol{e_{th}} = (\boldsymbol{e_t}||\boldsymbol{e_h})$, $\boldsymbol{c_i} = (\boldsymbol{c_i^h}||\boldsymbol{c_i^t})$, $\boldsymbol{s_i} = (\boldsymbol{s_i^t}||\boldsymbol{s_i^h})$, and $\boldsymbol{w_i} = (\boldsymbol{w_i^h}||\boldsymbol{w_i^t})$, then D is defined as:

$$D(h, r_i, t) = \begin{cases} \boldsymbol{\tau_{r_i(h,t)}} \oslash \boldsymbol{d_i}, & \text{if } \boldsymbol{\tau_{r_i(h,t)}} \preceq \boldsymbol{w_i} \\ \boldsymbol{\tau_{r_i(h,t)}} \odot \boldsymbol{d_i} - \boldsymbol{k_i}, & \text{otherwise} \end{cases} \tag{4.3}$$

Where $\boldsymbol{d_i} = \boldsymbol{2} \odot \boldsymbol{w_i} + \boldsymbol{1}$ is a width-dependent factor and $\boldsymbol{k_i} = \boldsymbol{0.5} \odot (\boldsymbol{d_i} - \boldsymbol{1}) \odot (\boldsymbol{d_i} - \boldsymbol{1} \oslash \boldsymbol{d_i})$. Observe that Equation 4.3 splits D into two parts:

- $D_i(h, r_i, t) = \boldsymbol{\tau_{r_i(h,t)}} \oslash \boldsymbol{d_i}$ for points inside the corresponding relation hyper-parallelogram, i.e., $\boldsymbol{\tau_{r_i(h,t)}} \preceq \boldsymbol{w_i}$.
- $D_o(h, r_i, t) = \boldsymbol{\tau_{r_i(h,t)}} \odot \boldsymbol{d_i} - \boldsymbol{k_i}$ for points outside the corresponding relation hyper-parallelogram, i.e., $\boldsymbol{\tau_{r_i(h,t)}} \not\preceq \boldsymbol{w_i}$.

Intuition. The general idea of splitting the distance function is to assign high scores to entity pair embeddings within a hyper-parallelogram and low scores to entity pair embeddings outside the hyper-parallelogram. Specifically, if a triple $r_i(h,t)$ is captured to be true by an ExpressivE embedding, i.e., if $\boldsymbol{\tau_{r_i(h,t)}} \preceq \boldsymbol{w_i}$, then the distance correlates inversely with the hyper-parallelogram's width — through the width-dependent factor $\boldsymbol{d_i}$ — keeping low distances/gradients for points within the hyper-parallelogram. Otherwise, the distance correlates — again through the width-dependent factor $\boldsymbol{d_i}$ — linearly with the width to penalize points outside larger parallelograms.

Scoring Function. Based on this distance function $D(h, r_i, t)$, we define ExpressivE's *scoring function* for quantifying the plausibility of a given triple $r_i(h,t)$ as follows:

$$s(h, r_i, t) = -||D(h, r_i, t)||_2 \tag{4.4}$$

Now that the ExpressivE model and its components have been introduced, we are ready to discuss its theoretical capabilities next.

4.2 Knowledge Capturing Capabilities

This section theoretically analyzes ExpressivE's expressive power and supported rules. As the proofs of our theorems are quite technical and long, we discuss the theorems and their general intuitions first (Sections 4.2.1 and 4.2.2), formalize important notions (Section 4.2.3) next, thereby, laying the theoretical foundations for the upcoming proofs (Sections 4.2.4–4.2.8). In what follows, we assume the standard definition of capturing rules (Sun et al., 2019; Abboud et al., 2020). This means intuitively that a KGE captures a rule if a set of parameters exists such that the rule is captured *exactly* and *exclusively*. We formalize this notion for ExpressivE in Section 4.2.3 to keep it close to our proofs.

4.2.1 Expressiveness

This section analyzes whether ExpressivE is *fully expressive* (Abboud et al., 2020), i.e., can capture any graph G over $\boldsymbol{R}$ and $\boldsymbol{E}$. Theorem 4.2.1 proves that this is the case by constructing for any graph G an ExpressivE embedding that captures any triple within G to be true and any other triple to be false. Specifically, the proof uses induction, starting with an embedding that captures the complete graph, i.e., any triple over $\boldsymbol{E}$ and $\boldsymbol{R}$ is true. Next, each induction step shows that we can alter the embedding to make an arbitrarily picked triple of the form $r_i(e_j, e_k)$ with $r_i \in \boldsymbol{R}$, $e_j, e_k \in \boldsymbol{E}$ and $e_j \neq e_k$ false. Finally, we add $|\boldsymbol{E}| * |\boldsymbol{R}|$ dimensions to make any self-loop — i.e., any triple of the form $r_i(e_j, e_j)$ with $r_i \in \boldsymbol{R}$ and $e_j \in \boldsymbol{E}$ — false. The full, quite technical proof can be found in Section 4.2.4.

Theorem 4.2.1 (Expressive Power)**.** *ExpressivE can capture any arbitrary graph G over $\boldsymbol{R}$ and $\boldsymbol{E}$ if the embedding dimensionality d is at least in $O(|\boldsymbol{E}| * |\boldsymbol{R}|)$.*

4.2.2 Inference Rules

This section shows that ExpressivE can capture all core inference rules (as can be seen in Table 1.1). First, we discuss how ExpressivE represents inference rules with at most two variables. Next, we introduce the notion of compositional definition and continue by identifying how this rule is described in the virtual triple space. Then, we define general composition, building on both the notion of compositional definition and hierarchy. Finally, we conclude this section by discussing the key properties of ExpressivE.

Two-Variable Rules. Figure 4.1b displays several one-dimensional relation embeddings and their captured rules in a correlation subspace. Intuitively, ExpressivE represents: (1) symmetry rules $r_1(X,Y) \Rightarrow r_1(Y,X)$ via symmetric hyper-parallelograms, (2) anti-symmetry rules $r_1(X,Y) \land r_1(Y,X) \Rightarrow \bot$ via hyper-parallelograms that do not overlap with their mirror image, (3) inversion rules $r_1(X,Y) \Leftrightarrow r_2(Y,X)$ via r_2's hyper-parallelogram being the mirror image of r_1's, (4) hierarchy rules $r_1(X,Y) \Rightarrow r_2(X,Y)$ via r_2's hyper-parallelogram subsuming r_1's, (5) intersection rules $r_1(X,Y) \land r_2(X,Y) \Rightarrow r_3(X,Y)$ via r_3's hyper-parallelogram subsuming the intersection of r_1's and r_2's, and (6) mutual exclusion rules $r_1(X,Y) \land r_2(X,Y) \Rightarrow \bot$ via mutually exclusive hyper-parallelograms of r_1 and r_2. We have formally proven that ExpressivE can capture any of these two-variable inference rules in Theorem 4.2.2 (see Sections 4.2.6 and 4.2.7).

Theorem 4.2.2. *ExpressivE captures (a) symmetry, (b) anti-symmetry, (c) inversion, (d) hierarchy, (e) intersection, and (f) mutual exclusion.*

Compositional Definition. A compositional definition rule is of the form $r_1(X,Y) \land r_2(Y,Z) \Leftrightarrow r_d(X,Z)$, where we call r_1 and r_2 the *composing* and r_d the *compositionally defined relation.* In essence, this rule defines a relation r_d that describes the start and end entities of a path $X \xrightarrow{r_1} Y \xrightarrow{r_2} Z$. Since any two relations r_1 and r_2 can instantiate the body of a compositional definition rule, any such pair may produce a new compositionally defined relation r_d. Interestingly, compositional definition rules translate analogously into the virtual triple space: Intuitively, this means that the embeddings of any two relations r_1 and r_2 define for r_d a *convex* region — which we call the *compositionally defined region* — that captures $r_1(X,Y) \land r_2(Y,Z) \Leftrightarrow r_d(X,Z)$, leading to Theorem 4.2.3 (proven in Section 4.2.5). Based on this insight, ExpressivE captures compositional definition rules by embedding the compositionally defined relation r_d with the compositionally defined region, defined by the relation embeddings of r_1 and r_2. We have formally proven that ExpressivE can capture compositional definition in Theorem 4.2.4 (see Sections 4.2.6 and 4.2.7 for the full proofs).

Theorem 4.2.3. *Let $r_1, r_2, r_d \in \boldsymbol{R}$ be relations, $\boldsymbol{A_1}, \boldsymbol{A_2}$ be their ExpressivE embeddings, and assume $r_1(X,Y) \land r_2(Y,Z) \Leftrightarrow r_d(X,Z)$ holds. Then there exists a region $\boldsymbol{A_d}$ in the virtual triple space $\mathbb{R}^{2d}$ such that (i) $\boldsymbol{A_1}, \boldsymbol{A_2}$, and $\boldsymbol{A_d}$ capture $r_1(X,Y) \land r_2(Y,Z) \Leftrightarrow r_d(X,Z)$ and (ii) $\boldsymbol{A_d}$ is* convex.

General Composition. In contrast to compositional definition, general composition $r_1(X,Y) \land r_2(Y,Z) \Rightarrow r_3(X,Z)$ does not specify the composed relation r_3 completely. Specifically, general composition allows the relation r_3 to include additional entity pairs not described by the start and end entities of the path $X \xrightarrow{r_1} Y \xrightarrow{r_2} Z$. Therefore, to capture general composition, we need to combine hierarchy and compositional definition. Formally, this means that we express general composition as: $\{r_1(X,Y) \land r_2(Y,Z) \Leftrightarrow r_d(X,Z),\ r_d(X,Y) \Rightarrow r_3(X,Y)\}$, where r_d is an auxiliary relation. We have proven that ExpressivE can capture general composition in Theorem 4.2.4 (see Sections 4.2.6 and 4.2.7 for the full proofs).

Theorem 4.2.4. *ExpressivE captures compositional definition and general composition.*

We argue that hierarchy and general composition are very tightly connected as hierarchies are hidden within general composition rules. If, for instance, r_1 were to represent the relation that solely captures self-loops, then the general composition rule $r_1(X,Y) \land r_2(Y,Z) \Rightarrow r_3(X,Z)$ would reduce to a hierarchy rule $r_2(X,Y) \Rightarrow r_3(X,Y)$. This hints at why our model is the first to support general composition, as ExpressivE can capture both hierarchy and composition jointly in a single embedding space. Finally, from Theorems 4.2.2 and 4.2.4 directly follows Corollary 4.2.5, which states that ExpressivE captures vital logical rules of the DB and SW community and, thus, reaches G1.

Corollary 4.2.5. *ExpressivE captures all core inference rules.*

Relation to OWL 2. Next, we briefly discuss the relation of OWL 2 axioms to the set of core inference rules and ExpressivE's limitations. Specifically, the OWL 2 standard (Motik et al., 2012c) defines the following eleven axioms between properties: (*i*) SymmetricObjectProperty, (*ii*) AsymmetricObjectProperty, (*iii*) InverseObjectProperties, (*iv*) SubObjectPropertyOf, (*v*) EquivalentObjectProperties, (*vi*) DisjointObjectProperties, (*vii*) FunctionalObjectProperty, (*viii*) InverseFunctionalObjectProperty, (*ix*) ReflexiveObjectProperty, (*x*) IrreflexiveObjectProperty, and (*xi*) TransitiveObjectProperty. Observe that the set of core inference rules already covers a large portion of these axioms. In particular, SymmetricObjectProperty axioms correspond to symmetry rules; AsymmetricObjectProperty axioms to antisymmetry rules; InverseObjectProperties axioms to inversion rules; SubObjectPropertyOf axioms to hierarchy rules; EquivalentObjectProperties axioms can be straightforwardly expressed by two hierarchy rules, stating that one relation subsumes the other and vice versa; and DisjointObjectProperties correspond to mutual exclusion rules.

Limitations. Whether ExpressivE captures the remaining OWL 2 axioms has not been analyzed yet; however, it is not likely to be the case. For instance, TransitiveObjectProperty axioms describe that a relation r is transitive, i.e., that $r(X,Y) \land r(Y,Z) \Rightarrow r(X,Z)$ holds. Recall that ExpressivE captures composition rules of the form $r_1(X,Y) \land r_2(Y,Z) \Rightarrow r_3(X,Z)$ by defining the embedding of r_3 as a superset of the compositionally defined region specified by the embeddings of r_1 and r_2. Thus, if all relations in the composition rule were the same — as is the case in transitivity rules — it would be necessary to find a fixed point such that the compositionally defined region specified by the relation embedding of r is again subsumed by r's embedding. This way of capturing transitive relations might be too restrictive, likely forcing r's hyper-parallelogram to be an axis-aligned square, which would instead represent a cross-product relation rather than a transitive one. Similarly, the constraints for the other missing axioms might be too restrictive for the ExpressivE model, demanding further research on developing a gKGE that can capture all axioms between properties of OWL 2. Still, ExpressivE captures additional rules not covered by OWL 2 axioms, such as general composition, compositional definition, and intersection rules, which have been vital for its KGC performance, as will be discussed in Section 4.5.

Key Properties. ExpressivE's way of capturing rules has several interesting implications:

1. We observe that ExpressivE embeddings offer an intuitive geometric interpretation: there is a natural correspondence between (a) relations in the KG – and – regions (representing mathematical relations) in the virtual triple space, (b) relation containment, intersection, and disjointness in the KG – and – region containment, intersection, and disjointness in the virtual triple space, (c) symmetry, anti-symmetry, and inversion in the KG – and – symmetry, anti-symmetry, and reflection in the virtual triple space, (d) compositional definition in the KG – and – the composition of mathematical relations in the virtual triple space.

2. Next, we observe that ExpressivE captures a general composition rule if the hyper-parallelogram of the rule's head relation subsumes the compositionally defined region defined by its body relations. Thereby, ExpressivE assigns a novel spatial interpretation to general composition rules, generalizing the spatial interpretation that is directly provided by rules describing set-theoretic properties such as hierarchy, intersection, and mutual exclusion.

3. Finally, capturing general composition rules through the subsumption of spatial regions allows ExpressivE to provably capture composition rules for *1–N*, *N–1*, and *N–N* relations. We provide further empirical evidence to this in Section 4.5.4.

4.2.3 Formal Definitions

In this section, we formally introduce the notions of capturing a rule in an ExpressivE model that we informally discussed in Section 4.2. Furthermore, we will introduce some additional notations, which will help us simplify the upcoming proofs and present them intuitively. Let us start by summarizing the notation used throughout this chapter:

v . . . non-bold symbols represent scalars

$\boldsymbol{v}$ . . . bold symbols represent vectors, sets or tuples

$\mathbf{0}$ . . . represents a vector of solely zeros (the same semantics apply to $\mathbf{0.5}$, $\mathbf{1}$, and $\mathbf{2}$)

$\oslash$ . . . represents the element-wise division operator

$\odot$ . . . represents the element-wise (Hadamard) product operator

$\succeq$ . . . represents the element-wise greater or equal operator

$\succ$ . . . represents the element-wise greater operator

$\preceq$ . . . represents the element-wise less or equal operator

$\prec$ . . . represents the element-wise less operator

$\boldsymbol{x}^{|.|}$ . . . represents the element-wise absolute value

$||$ . . . represents the concatenation operator

$\boldsymbol{v}(j)$ . . . represents the j-th dimension of a vector $\boldsymbol{v}$

Knowledge Graph. Formally, a tuple $(\boldsymbol{G}, \boldsymbol{E}, \boldsymbol{R})$ is called a Knowledge Graph, where $\boldsymbol{R}$ is a finite set of relations, $\boldsymbol{E}$ is a finite set of entities, and $\boldsymbol{G} \subseteq \boldsymbol{E} \times \boldsymbol{R} \times \boldsymbol{E}$ is a finite set of triples. W.l.o.g., we assume that any relation is non-empty since assigning an empty hyper-parallelogram to an empty relation would be trivial, just adding unnecessary complexity to the proofs.

ExpressivE model. A tuple $\boldsymbol{M} = (\boldsymbol{\epsilon}, \boldsymbol{\sigma}, \boldsymbol{\delta}, \boldsymbol{\rho})$ is called an ExpressivE model, where $\boldsymbol{\epsilon} \subset 2^{\mathbb{R}^d}$ is a finite set of entity embeddings, $\boldsymbol{\sigma} \subset 2^{\mathbb{R}^d}$ is a finite set of center embeddings, $\boldsymbol{\delta} \subset 2^{\mathbb{R}^d}$ is a finite set of width embeddings, and $\boldsymbol{\rho} \subset 2^{\mathbb{R}^d}$ is a finite set of slope vectors.

Linking Embeddings to KGs. An ExpressivE model and a KG are linked via the following assignment functions: The entity assignment function $\boldsymbol{f_e} : \boldsymbol{E} \rightarrow \boldsymbol{\epsilon}$ assigns an entity embedding $\boldsymbol{e_h} \in \boldsymbol{\epsilon}$ to each entity $e_h \in \boldsymbol{E}$. Based on $\boldsymbol{f_e}$, the virtual assignment function $\boldsymbol{f_v} : \boldsymbol{E} \times \boldsymbol{E} \rightarrow \mathbb{R}^{2d}$ defines for any pair of entities $(e_h, e_t) \in \boldsymbol{E}$ a virtual entity pair embedding $\boldsymbol{f_v}(e_h, e_t) = (\boldsymbol{f_e}(e_h)||\boldsymbol{f_e}(e_t))$, where $||$ represents the concatenation operator. Furthermore, the relation assignment function $\boldsymbol{f_h}(r_i) : \boldsymbol{R} \rightarrow \mathbb{R}^{2d} \times \mathbb{R}^{2d} \times \mathbb{R}^{2d}$

assigns a hyper-parallelogram to each relation r_i. In more detail, this means that $\boldsymbol{f_h}(r_i) = (\boldsymbol{c_i}, \boldsymbol{w_i}, \boldsymbol{s_i})$, where $\boldsymbol{c_i} = (\boldsymbol{c_i^h} || \boldsymbol{c_i^t})$ are two concatenated center embeddings with $\boldsymbol{c_i^h}, \boldsymbol{c_i^t} \in \boldsymbol{\sigma}$, where $\boldsymbol{w_i} = (\boldsymbol{w_i^h} || \boldsymbol{w_i^t})$ are two concatenated width embeddings with $\boldsymbol{w_i^h}, \boldsymbol{w_i^t} \in \boldsymbol{\delta}$, and where $\boldsymbol{s_i} = (\boldsymbol{s_i^t} || \boldsymbol{s_i^h})$ are two concatenated slope vectors with $\boldsymbol{s_i^t}, \boldsymbol{s_i^h} \in \boldsymbol{\rho}$. Intuitively, $\boldsymbol{f_h}(r_i)$ defines a hyper-parallelogram in the virtual triple space $\mathbb{R}^{2d}$ as described in Section 4.1.

Model Configuration. We call an ExpressivE model $\boldsymbol{M}$ together with a concrete relation assignment function $\boldsymbol{f_h}$ a relation configuration $\boldsymbol{m_h} = (\boldsymbol{M}, \boldsymbol{f_h})$ and if it additionally has a concrete virtual assignment function $\boldsymbol{f_v}$, we call it a complete model configuration $\boldsymbol{m} = (\boldsymbol{M}, \boldsymbol{f_h}, \boldsymbol{f_v})$.

Definition of Truth. A triple $r_i(e_h, e_t)$ holds in some $\boldsymbol{m}$, with $r_i \in \boldsymbol{R}$ and $e_h, e_t \in \boldsymbol{E}$ iff Inequalities 4.1 and 4.2 hold for the assigned embeddings of h, t, and r. This means more specifically that Inequalities 4.1 and 4.2 need to hold for $\boldsymbol{f_v}(e_h, e_t) = (\boldsymbol{f_e}(e_h) || \boldsymbol{f_e}(e_t)) = (\boldsymbol{e_h} || \boldsymbol{e_t})$ and $\boldsymbol{f_h}(r_i) = (\boldsymbol{c_i}, \boldsymbol{w_i}, \boldsymbol{s_i})$, with $\boldsymbol{c_i} = (\boldsymbol{c_i^h} || \boldsymbol{c_i^t})$, $\boldsymbol{w_i} = (\boldsymbol{w_i^h} || \boldsymbol{w_i^t})$, and $\boldsymbol{s_i} = (\boldsymbol{s_i^t} || \boldsymbol{s_i^h})$. At an intuitive level, this means that a triple $r_i(e_h, e_t)$ is true in some complete model configuration $\boldsymbol{m}$ iff the virtual pair embedding $\boldsymbol{f_v}(e_h, e_t)$ of entities e_h and e_t lies within the hyper-parallelogram of relation r_i defined by $\boldsymbol{f_h}(r_i)$.

Simplifying Notations. Therefore, to simplify the upcoming proofs, we denote with $\boldsymbol{f_v}(e_h, e_t) \in \boldsymbol{f_h}(r_i)$ that the virtual pair embedding $\boldsymbol{f_v}(e_h, e_t) \in \mathbb{R}^{2d}$ of an entity pair $(e_h, e_t) \in \boldsymbol{E} \times \boldsymbol{E}$ lies within the hyper-parallelogram $\boldsymbol{f_h}(r_i) \subseteq \mathbb{R}^{2d} \times \mathbb{R}^{2d} \times \mathbb{R}^{2d}$ of some relation $r_i \in \boldsymbol{R}$ in the virtual triple space. Accordingly, for sets of virtual pair embeddings $\boldsymbol{P} := \{\boldsymbol{f_v}(e_{h_1}, e_{t_1}), \dots, \boldsymbol{f_v}(e_{h_n}, e_{t_n})\}$, we denote with $\boldsymbol{P} \subseteq \boldsymbol{f_h}(r_i)$ that all virtual pair embeddings of $\boldsymbol{P}$ lie within the hyper-parallelogram of the relation r_i. Furthermore, we denote with $\boldsymbol{f_v}(e_h, e_t) \notin \boldsymbol{f_h}(r_i)$ that a virtual pair embedding $\boldsymbol{f_v}(e_h, e_t)$ does not lie within the hyper-parallelogram of a relation r_i and with $\boldsymbol{P} \not\subseteq \boldsymbol{f_h}(r_i)$ we denote that an entire set of virtual pair embeddings $\boldsymbol{P}$ does not lie within the hyper-parallelogram of a relation r_i.

Capturing Inference Rules. Based on the previous definitions, we define capturing rules formally: A relation configuration $\boldsymbol{m_h}$ captures a rule ψ *exactly* if for any ground rule $\phi_{B_1} \wedge \dots \wedge \phi_{B_m} \Rightarrow \phi_H$ within the deductive closure of ψ and for any instantiation of $\boldsymbol{f_e}$ and $\boldsymbol{f_v}$ the following conditions are satisfied:

- if ϕ_H is a triple and if $\boldsymbol{m_h}$ captures the body triples to be true — i.e., $\boldsymbol{f_v}(\mathit{args}(\phi_{B_1})) \in \boldsymbol{f_h}(\mathit{rel}(\phi_{B_1})), \dots, \boldsymbol{f_v}(\mathit{args}(\phi_{B_m})) \in \boldsymbol{f_h}(\mathit{rel}(\phi_{B_m}))$ — then $\boldsymbol{m_h}$ also captures the head triple to be true — i.e., $\boldsymbol{f_v}(\mathit{args}(\phi_H)) \in \boldsymbol{f_h}(\mathit{rel}(\phi_H))$.
- if $\phi_H = \perp$, then $\boldsymbol{m_h}$ captures at least one of the body triples to be false — i.e., there is some $j \in \{1, \dots, m\}$ such that $\boldsymbol{f_v}(\mathit{args}(\phi_{B_j})) \notin \boldsymbol{f_h}(\mathit{rel}(\phi_{B_j}))$.

where *args*() is the function that returns the arguments of a triple and *rel*() is the function that returns the relation of the triple. Furthermore, a relation configuration $\boldsymbol{m_h}$ captures a rule ψ *exactly* and *exclusively* if (1) $\boldsymbol{m_h}$ exactly captures ψ and (2) $\boldsymbol{m_h}$ does not capture any *positive* rule ϕ (i.e., $\phi \in \{symmetry,\ inversion,\ hierarchy,\ intersection,\ composition\}$) such that $\psi \not\models \phi$ except where the body of ϕ is not satisfied over $\boldsymbol{m_h}$.

Discussion. Next, the intuition of the above definition of capturing a rule is discussed.

Capturing a rule *exactly* is defined straightforwardly by adhering to the semantics of logical implication $\phi := \phi_B \Rightarrow \phi_H$, i.e., a relation configuration $\boldsymbol{m_h}$ needs to be found such that for any complete model configuration $\boldsymbol{m}$ over $\boldsymbol{m_h}$ if the body ϕ_B of the rule is satisfied, then its head ϕ_H can be inferred.

Capturing a rule *exactly* and *exclusively* imposes additional constraints. Here, the aim is not solely to capture a rule but additionally to showcase that a rule can be captured independently from any other rule. Therefore, some notion of minimality/exclusiveness of a rule is needed. As in Abboud et al. (2020), we define minimality by means of *solely* capturing those positive rules ϕ that directly follow from the deductive closure of the rule ψ, except for those ϕ that are captured trivially, i.e., except for those ϕ where their body is not satisfied over the constructed $\boldsymbol{m_h}$.

As presented in Section 4.2, we can express any core inference rule (defined in Section 2) by means of spatial relations of the corresponding relation hyper-parallelograms in the virtual triple space. Therefore, we formulate *exclusiveness* intuitively as the ability to limit the intersection of hyper-parallelograms to only those intersections that directly follow from the captured rule ψ for any known relation $r_i \in \boldsymbol{R}$, which is in accordance with BoxE's notion of exclusiveness (Abboud et al., 2020).

Note that our definition of capturing rules solely depends on relation configurations. This is vital for ExpressivE to be able to capture rules in a *lifted* manner, i.e., ExpressivE shall be able to capture rules without the need of grounding them first. Furthermore, being able to capture rules in a lifted way is not only efficient but also natural as we aim at capturing rules between relations. Thus it would be unnatural if constraints on entity embeddings were necessary to capture such relation-specific rules.

As outlined in the previous paragraphs, our definition of capturing rules is in accordance with the literature (Abboud et al., 2020), focuses on efficiently capturing rules, and gives us a formal foundation for the upcoming proofs, which will show that ExpressivE is fully expressive (Section 4.2.4), can capture all core inference rules (Sections 4.2.6 and 4.2.7), and is not limited to solely capturing a single composition rule (Section 4.2.8).

4.2.4 Proof of Fully Expressiveness (Theorem 4.2.1)

In this section, we prove Theorem 4.2.1. We will show by induction that ExpressivE is fully expressive. We will first only consider self-loop-free triples, i.e., triples of the form $r_i(e_j, e_k)$ with $e_j, e_k \in \boldsymbol{E}$, $r_i \in \boldsymbol{R}$ and $j \neq k$ and later remove unwanted self-loops from the constructed model configuration.

Since our proof is highly technical, we will first give some general intuition and then formally state our proof. In the base case, we consider an ExpressivE model that captures the complete graph G over the entity vocabulary $\boldsymbol{E}$ and the relationship vocabulary $\boldsymbol{R}$, i.e., the graph that contains all triples from the universe. In the induction step, we prove that we can adjust our ExpressivE model to make any arbitrary self-loop-free triple of G false while maintaining the truth value of any other triple in the universe.

In the induction step, we make triples $r_i(e_j, e_k)$ false by translating the entity embeddings of e_j and e_k such that a hyper-parallelogram can separate pairs of entity embeddings that shall be true from those that shall be false. Afterward, we translate and shear r_i's hyper-parallelogram to match such a separating shape.

Finally, after the induction step, we add a separate dimension for any possible self-loop, i.e., triple of the form $r_i(e_j, e_j)$, to make them false. Thereby, we show that ExpressivE can make any triple false and thus that it can capture any graph G over $\boldsymbol{R}$ and $\boldsymbol{E}$.

Our proof shares some common ideas with the fully expressiveness proof of BoxE (Abboud et al., 2020), yet differs dramatically in many aspects. BoxE embeds relations with two axis-aligned boxes and entities with two separate embedding vectors, which greatly simplifies the fully expressiveness proof of BoxE, as the two entity embeddings are independent of each other. This grants BoxE some flexibility for adapting model configuration yet imposes substantial restrictions, such as BoxE not being able to capture any notion of composition rules. Our model does not have these restrictions and uses only one embedding vector per entity instead, pushing the complexity of our model to the relation embeddings by representing relations as hyper-parallelogram in the virtual triple space. This, however, has the consequence that we cannot easily change entity embeddings without moving and sheering relation embeddings as well when we want to make solely one triple false and preserve the truth value of any other triple. In the following proof, we will explain the complex adjustment of relation embeddings and many more novel aspects of our proof in more detail.

Proof. We start our proof by making the following assumptions without loss of generality:

1. Any relation $r_i \in \boldsymbol{R}$ and entity $e_j \in \boldsymbol{E}$ is indexed with $0 \leq i \leq |\boldsymbol{R}| - 1$ and $0 \leq j \leq |\boldsymbol{E}| - 1$.
2. The dimensionality of each relation and entity embedding vectors is equal to $|\boldsymbol{E}| * |\boldsymbol{R}|$. Furthermore, $\boldsymbol{v}(i, j)$ represents the dimension $i * |\boldsymbol{E}| + j$ of the vector $\boldsymbol{v}$. Intuitively, the dimensions of $\boldsymbol{v}(i, 0), \ldots, \boldsymbol{v}(i, |\boldsymbol{E}| - 1)$ corresponds to the dimensions reserved for relation r_i.
3. The slope vectors of relation $r_i \in \boldsymbol{R}$ are positive, i.e., $\boldsymbol{s_i^h}, \boldsymbol{s_i^t} > 0$.
4. Any entity embedding is positive, i.e., for any entity $e_k \in \boldsymbol{E}$ holds that $\boldsymbol{e_k} > 0$.
5. For any pair of entities $e_{k_1}, e_{k_2} \in \boldsymbol{E}$ holds that $\boldsymbol{e_{k_1}}(i, k_1) \geq \boldsymbol{e_{k_2}}(i, k_1) + m$, with $m > 0$.

Building on these assumptions, we prove fully expressiveness by induction as follows:

Base Case. We initialize a graph G as the whole universe over $\boldsymbol{E}$ and $\boldsymbol{R}$ and construct a complete model configuration $\boldsymbol{m} = (\boldsymbol{M}, \boldsymbol{f_h}, \boldsymbol{f_v})$ with dimensionality $|\boldsymbol{E}| * |\boldsymbol{R}|$ such that G is captured and all assumptions are satisfied. Concretely, we specify for any dimension (i, k_1) with $0 \leq i \leq |\boldsymbol{R}| - 1$ and $0 \leq k_1 \leq |\boldsymbol{E}| - 1$ the embedding values of entity embeddings with index k_1 to set $\boldsymbol{e_{k_1}}(i, k_1) = 2$ and with index $k_2 \neq k_1$ to $\boldsymbol{e_{k_2}}(i, k_1) = 1$. Furthermore, we specify for any dimension (i, k) with $0 \leq i \leq |\boldsymbol{R}| - 1$ and $0 \leq k \leq |\boldsymbol{E}| - 1$ the embedding of relation r_i to $\boldsymbol{c_i^h}(i, k) = \boldsymbol{c_i^t}(i, k) = 0$, $\boldsymbol{s_i^h}(i, k) = 1$, $\boldsymbol{s_i^t}(i, k) = 2$ and $\boldsymbol{w_i^h}(i, k) = \boldsymbol{w_i^t}(i, k) = 4$. As can be shown easily, the constructed complete model configuration satisfies all assumptions and makes any triple over $\boldsymbol{R}$ and $\boldsymbol{E}$ true. Note that, in particular, any self-loop is also captured to be true in the constructed complete model configuration.

Induction Step. In the induction step, we adjust the entity and relation embeddings of the complete model configuration such that a single triple $r_i(e_j, e_k)$ is made false without affecting the truth value of any other triple within the graph G. We denote any adjusted embedding with an asterisk $\boldsymbol{v}^*$ and the old value of the embedding with $\boldsymbol{v}$ and perform the following adjustments:

1. Increase any slope vector $\boldsymbol{s_i^{t*}}(i, k) := \boldsymbol{s_i^t}(i, k) + \Delta r_i^t$ with $\Delta r_i^t > 0$ such that:

$$\boldsymbol{e_j}(i, k) - \boldsymbol{s_i^t}(i, k)\boldsymbol{e_k}(i, k) - \boldsymbol{c_i^h}(i, k) - \Delta r_i^t m \leq -\boldsymbol{w_i^h}(i, k)$$

2. Since $\boldsymbol{e_k}(i, k)$ is by assumption the largest value in dimension (i, k), we can specify the following two values:

$$\Delta r_i^{max} := \Delta r_i^t \boldsymbol{e_k}(i, k)$$

$$\Delta r_i^{ub} := \Delta r_i^t (\boldsymbol{e_k}(i, k) - m)$$

   with $\Delta r_i^{ub} < \Delta r_i^{max}$.

3. Using this definition, we increase all entity embeddings $\boldsymbol{e_{j'}}$ with $j' \neq j$ in dimension (i, k) by:

$$\boldsymbol{e_{j'}^*}(i, k) := \boldsymbol{e_{j'}}(i, k) + \Delta r_i^{max}$$

4. Furthermore, we increase the entity embedding $\boldsymbol{e_j}$ in dimension (i, k) by:

$$\boldsymbol{e_j^*}(i, k) := \boldsymbol{e_j}(i, k) + \Delta r_i^{ub}$$

5. For any relation with index $i \neq i'$, we adjust any head band in dimension (i, k) by moving its center downwards and growing the band upwards. This means formally that we update the following embeddings:

$$\begin{aligned} s &:= \boldsymbol{s}^{\boldsymbol{t}}_{\boldsymbol{i'}}(i,k)\Delta r^t_i m + \Delta r^{max}_i \\ \boldsymbol{w}^{\boldsymbol{h}*}_{\boldsymbol{i'}}(i,k) &:= \boldsymbol{w}^{\boldsymbol{h}}_{\boldsymbol{i'}}(i,k) + \frac{s}{2} \\ \boldsymbol{c}^{\boldsymbol{h}*}_{\boldsymbol{i'}}(i,k) &:= \boldsymbol{c}^{\boldsymbol{h}}_{\boldsymbol{i'}}(i,k) - \boldsymbol{s}^{t}_{\boldsymbol{i'}}(i,k)\Delta r^{max}_i + \frac{s}{2} \end{aligned}$$

6. We adjust any tail band in dimension (i, k) by moving its center downwards and growing the band upwards. This means formally that we update the following embeddings:

$$\begin{aligned} s &:= \boldsymbol{s}^{\boldsymbol{h}}_{\boldsymbol{i'}}(i,k)\Delta r^t_i m + \Delta r^{max}_i \\ \boldsymbol{w}^{\boldsymbol{t}*}_{\boldsymbol{i'}}(i,k) &:= \boldsymbol{w}^{\boldsymbol{t}}_{\boldsymbol{i'}}(i,k) + \frac{s}{2} \\ \boldsymbol{c}^{\boldsymbol{t}*}_{\boldsymbol{i'}}(i,k) &:= \boldsymbol{c}^{\boldsymbol{t}}_{\boldsymbol{i'}}(i,k) - \boldsymbol{s}^{h}_{\boldsymbol{i'}}(i,k)\Delta r^{max}_i + \frac{s}{2} \end{aligned}$$

7. For any relation with index i, we adjust any head band in dimension (i, k) by moving its center downwards and growing the band upwards. This means formally that we update the following embeddings:

$$\begin{aligned} s &:= (\Delta r^t_i + \boldsymbol{s}^{\boldsymbol{t}}_{\boldsymbol{i}}(i,k))\Delta r^t_i m + \Delta r^{max}_i \\ \boldsymbol{w}^{\boldsymbol{h}*}_{\boldsymbol{i}}(i,k) &:= \boldsymbol{w}^{\boldsymbol{h}}_{\boldsymbol{i}}(i,k) + \frac{s}{2} \\ \boldsymbol{c}^{\boldsymbol{h}*}_{\boldsymbol{i}}(i,k) &:= \boldsymbol{c}^{\boldsymbol{h}}_{\boldsymbol{i}}(i,k) - \Delta r^t_i \Delta r^{max}_i - \boldsymbol{s}^{t}_{\boldsymbol{i}}(i,k)\Delta r^{max}_i + \frac{s}{2} \end{aligned}$$

In the induction step, we adjust the slope vectors (Step 1), the entity embeddings (Step 2–4), and the width and center embeddings (Step 5–7). Intuitively, by changing the slope vector of relation hyper-parallelograms, we sheer the hyper-parallelograms. Furthermore, we translate any desired entity embeddings more than the undesired entity embedding of e_j. This allows us to draw a separating hyper-parallelogram between the point defined by (e_j, e_k) and any other pair of entities that shall remain within relation r_i. Finally, we must move the sheered hyper-parallelograms into the correct position and stretch it to make all desired triples true.

Our next goal is to show this behavior formally. We will first show that the initially true triple $r_i(e_j, e_k)$ is false, then continue by showing that the truth value of any other triple is preserved.

Since the induction steps perform only adjustments in dimension (i, k), we only have to consider the dimension (i, k) for any embedding vector in the following inequalities. Please note that to state the inequalities concisely, we omitted the notation (i, k) from any embedding vector $\boldsymbol{v}$ in the following inequalities. For instance, we will denote $\boldsymbol{s_i^t}(i, k)$ with $\boldsymbol{s_i^t}$ henceforth.

Let $s := (\Delta r_i^t + \boldsymbol{s_i^t})\Delta r_i^t m + \Delta r_i^{max}$, then we can show that our induction step makes $r_i(e_j, e_k)$ false as follows:

$$\boldsymbol{e_j} - \boldsymbol{s_i^t}\boldsymbol{e_k} - \boldsymbol{c_i^h} - \Delta r_i^t m \leq -\boldsymbol{w_i^h} \tag{4.5}$$

$$\begin{aligned} \boldsymbol{e_j} - \boldsymbol{s_i^t}\boldsymbol{e_k} - \boldsymbol{c_i^h} + \Delta r_i^{ub} - \Delta r_i^{max} - \Delta r_i^t \Delta r_i^{max} + \Delta r_i^t \Delta r_i^{max} \\ - \boldsymbol{s_i^t}\Delta r_i^{max} + \boldsymbol{s_i^t}\Delta r_i^{max} + \frac{s}{2} - \frac{s}{2} \leq -\boldsymbol{w_i^h} \end{aligned} \tag{4.6}$$

$$\begin{aligned} \boldsymbol{e_j} + \Delta r_i^{ub} - (\boldsymbol{s_i^t} + \Delta r_i^t)(\boldsymbol{e_k} + \Delta r_i^{max}) - (\boldsymbol{c_i^h} - \Delta r_i^t \Delta r_i^{max} \\ - \boldsymbol{s_i^t}\Delta r_i^{max} + \frac{s}{2}) \leq -(\boldsymbol{w_i^h} + \frac{s}{2}) \end{aligned} \tag{4.7}$$

$$\boldsymbol{e_j^*} - \boldsymbol{s_i^{t*}}\boldsymbol{e_k^*} - \boldsymbol{c_i^{h*}} \leq -\boldsymbol{w_i^{h*}} \tag{4.8}$$

Inequality 4.5 follows directly from Induction Step 1. Next, in Inequality 4.6 we add many terms that eliminate each other and apply $\Delta r_i^{ub} - \Delta r_i^{max} = \Delta r_i^t(\boldsymbol{e_k} - m) - \Delta r_i^t \boldsymbol{e_k} = -m\Delta r_i^t$. Finally, in Inequality 4.7, we restructure the terms such that we can substitute the terms for the adjusted embedding vectors defined in Steps 1–7. Through this substitution, we obtain Inequality 4.8, which reveals that the adjusted embeddings $\boldsymbol{e_j^*}, \boldsymbol{e_k^*}$ do not lie within the adjusted hyper-parallelogram of relation r_i. Therefore, we have shown that the adjustments of the complete model configuration listed in Steps 1–7 have made the triple $r_i(e_j, e_k)$ false, as required.

Next, we need to show that the truth value of any other self-loop-free triple $r_{i'}(e_{j'}, e_{k'})$ with $j' \neq k'$ is not altered after the induction step. We start by showing that any triple $r_{i'}(e_{j'}, e_{k'})$ that is true in $\boldsymbol{m}$ remains true after the induction step. Since what follows is a highly technical proof, we give some intuition now. We make a case distinction of any possible true triple in G and perform the following steps. First, we assume that the triple is true and therefore instantiate Inequalities 4.1 and 4.2 with the embeddings prior to the induction step. Note that it is solely necessary to consider Inequality 4.1 as the proofs work vice versa for Inequality 4.2. Thus, we solely consider Inequality 4.1 henceforth. Next, we add terms that eliminate each other and adjustment terms a such that we can substitute our inequality with the adjusted embedding values $\boldsymbol{v*}$. Finally, we show that Inequality 4.1 is satisfied for the adjusted embedding values. Note that Inequality 4.1 defines two inequalities, specifically $\boldsymbol{e_h} - \boldsymbol{c_i^h} - \boldsymbol{s_i^t} \odot \boldsymbol{e_t} \preceq \boldsymbol{w_i^h}$ and $\boldsymbol{e_h} - \boldsymbol{c_i^h} - \boldsymbol{s_i^t} \odot \boldsymbol{e_t} \succeq -\boldsymbol{w_i^h}$. Therefore, we denote with **(<)** the proof for the first inequality and with **(>)** the proof for the second inequality. Thereby, we will show that if we assume the triple $r_{i'}(e_{j'}, e_{k'})$ to be true in the complete model configuration prior to the induction step, we can follow that $r_{i'}(e_{j'}, e_{k'})$ stays true after the adjustments of the induction step. To provide the complete formal side of our proof, we consider the following 12 cases:

1. **Case** $i' = i, j' = j, k' = j, k' \neq k$:

   (<) Let $s := (\Delta r_i^t + \boldsymbol{s_i^t})\Delta r_i^t m + \Delta r_i^{max}$ and let $a := (\Delta r_i^{max} - \Delta r_i^{ub})(1 - \Delta r_i^t - \boldsymbol{s_i^t}\Delta r^{ub})$. Note that a is positive since $a = \Delta r_i^t m + \Delta r_i^{max}$ holds. Therefore, we can perform the following transformations:

$$\boldsymbol{e_j} - \boldsymbol{s_i^t e_j} - \boldsymbol{c_i^h} \leq \boldsymbol{w_i^h} \tag{4.9}$$

$$\boldsymbol{e_j} - \boldsymbol{s_i^t e_j} - \boldsymbol{c_i^h} - a + s - s \leq \boldsymbol{w_i^h} \tag{4.10}$$

$$\begin{aligned} \boldsymbol{e_j} + \Delta r_i^{ub} - (\boldsymbol{s_i^t} + \Delta r_i^t)(\boldsymbol{e_j} + \Delta r_i^{ub}) - (\boldsymbol{c_i^h} - \Delta r_i^t \Delta r_i^{max} \\ - \boldsymbol{s_i^t}\Delta r_i^{max} + \frac{s}{2}) \leq \boldsymbol{w_i^h} + \frac{s}{2} \end{aligned} \tag{4.11}$$

$$\boldsymbol{e_j^*} - \boldsymbol{s_i^{t*} e_j^*} - \boldsymbol{c_i^{h*}} \leq \boldsymbol{w_i^{h*}} \tag{4.12}$$

   (>) Let $a := (\Delta r_i^{max} - \Delta r_i^{ub})(\Delta r_i^t + \boldsymbol{s_i^t}) + \Delta r_i^{ub} - \Delta r_i^{max}$ and let $s := (\Delta r_i^t + \boldsymbol{s_i^t})\Delta r_i^t m + \Delta r_i^{max}$. Note that a is positive since (1) $a = m\Delta r_i^t(\Delta r_i^t + \boldsymbol{s_i^t} - 1)$, (2) we initialize $\boldsymbol{s_i^t}$ in the base case to 2 in any dimension and (3) any induction step may only increase $\boldsymbol{s_i^t}$. Therefore, we can perform the following transformations:

$$\boldsymbol{e_j} - \boldsymbol{s_i^t e_j} - \boldsymbol{c_i^h} \geq -\boldsymbol{w_i^h} \tag{4.13}$$

$$\boldsymbol{e_j} - \boldsymbol{s_i^t e_j} - \boldsymbol{c_i^h} + a + \frac{s}{2} - \frac{s}{2} \geq -\boldsymbol{w_i^h} \tag{4.14}$$

$$\begin{aligned} \boldsymbol{e_j} + \Delta r_i^{ub} - (\boldsymbol{s_i^t} + \Delta r_i^t)(\boldsymbol{e_j} + \Delta r_i^{ub}) - (\boldsymbol{c_i^h} - \Delta r_i^t \Delta r_i^{max} \\ - \boldsymbol{s_i^t}\Delta r_i^{max} + \frac{s}{2}) \geq -(\boldsymbol{w_i^h} + \frac{s}{2}) \end{aligned} \tag{4.15}$$

$$\boldsymbol{e_j^*} - \boldsymbol{s_i^{t*} e_j^*} - \boldsymbol{c_i^{h*}} \geq -\boldsymbol{w_i^{h*}} \tag{4.16}$$

2. **Case** $i' = i, j' = j, k' \neq j, k' = k$:

   As can be seen easily this case describes the triple $r_i(e_j, e_k)$, which shall be made false in the induction step. We have shown that the induction step changes the triples truth value to false in Inequalities 4.5–4.8 and therefore omitted the case here.

3. **Case** $i' = i, j' = j, k' \neq j, k' \neq k$:

   (<) Let $s := (\Delta r_i^t + \boldsymbol{s_i^t})\Delta r_i^t m + \Delta r_i^{max}$ and let $a := \Delta r_i^t \boldsymbol{e_{k'}} + s - \Delta r_i^{ub}$. Note that a is positive since $a = \Delta r_i^t(\boldsymbol{e_{k'}} + m(1 + \Delta r_i^t + \boldsymbol{s_i^t}))$ holds. Therefore, we can perform the following transformations:

$$\boldsymbol{e_j} - \boldsymbol{s_i^t}\boldsymbol{e_{k'}} - \boldsymbol{c_i^h} \leq \boldsymbol{w_i^h} \tag{4.17}$$

$$\boldsymbol{e_j} - \boldsymbol{s_i^t}\boldsymbol{e_{k'}} - \boldsymbol{c_i^h} - a + \Delta r_i^t \Delta r_i^{max} - \Delta r_i^t \Delta r_i^{max} + \boldsymbol{s_i^t}\Delta r_i^{max} - \boldsymbol{s_i^t}\Delta r_i^{max} \leq \boldsymbol{w_i^h} \tag{4.18}$$

$$\begin{aligned}\boldsymbol{e_j} + \Delta r_i^{ub} - (\boldsymbol{s_i^t} + \Delta r_i^t)(\boldsymbol{e_{k'}} + \Delta r_i^{max}) - (\boldsymbol{c_i^h} - \Delta r_i^t \Delta r_i^{max} \\ -\boldsymbol{s_i^t}\Delta r_i^{max} + \frac{s}{2}) \leq \boldsymbol{w_i^h} + \frac{s}{2}\end{aligned} \tag{4.19}$$

$$\boldsymbol{e_j^*} - \boldsymbol{s_i^{t*}}\boldsymbol{e_{k'}^*} - \boldsymbol{c_i^{h*}} \leq \boldsymbol{w_i^{h*}} \tag{4.20}$$

(>) Let $a := \Delta r_i^{ub} - \Delta r_i^t \boldsymbol{e_{k'}}$ and let $s := (\Delta r_i^t + \boldsymbol{s_i^t})\Delta r_i^t m + \Delta r_i^{max}$. Note that a is positive since $\Delta r_i^{ub} \geq \Delta r_i^t \boldsymbol{e_{k'}}$ holds. Therefore, we can perform the following transformations:

$$\boldsymbol{e_j} - \boldsymbol{s_i^t}\boldsymbol{e_{k'}} - \boldsymbol{c_i^h} \geq -\boldsymbol{w_i^h} \tag{4.21}$$

$$\begin{aligned}\boldsymbol{e_j} - \boldsymbol{s_i^t}\boldsymbol{e_{k'}} - \boldsymbol{c_i^h} + a + \Delta r_i^t \Delta r_i^{max} - \Delta r_i^t \Delta r_i^{max} + \boldsymbol{s_i^t}\Delta r_i^{max} \\ -\boldsymbol{s_i^t}\Delta r_i^{max} + \frac{s}{2} - \frac{s}{2} \geq -\boldsymbol{w_i^h}\end{aligned} \tag{4.22}$$

$$\begin{aligned}\boldsymbol{e_j} + \Delta r_i^{ub} - (\boldsymbol{s_i^t} + \Delta r_i^t)(\boldsymbol{e_{k'}} + \Delta r_i^{max}) - (\boldsymbol{c_i^h} - \Delta r_i^t \Delta r_i^{max} \\ -\boldsymbol{s_i^t}\Delta r_i^{max} + \frac{s}{2}) \geq -(\boldsymbol{w_i^h} + \frac{s}{2})\end{aligned} \tag{4.23}$$

$$\boldsymbol{e_j^*} - \boldsymbol{s_i^{t*}}\boldsymbol{e_{k'}^*} - \boldsymbol{c_i^{h*}} \geq -\boldsymbol{w_i^{h*}} \tag{4.24}$$

4. **Case** $i' = i, j' \neq j, k' = j, k' \neq k$:

(<) Let $a := \Delta r_i^t \boldsymbol{e_j}$ and let $s := (\Delta r_i^t + \boldsymbol{s_i^t})\Delta r_i^t m + \Delta r_i^{max}$. Note that a is trivially positive since we initially assumed $\boldsymbol{e_j} > 0$ and since we assumed $\Delta r_i^t > 0$ in Step 1. Therefore, we can perform the following transformations:

$$\boldsymbol{e_{j'}} - \boldsymbol{s_i^t}\boldsymbol{e_j} - \boldsymbol{c_i^h} \leq \boldsymbol{w_i^h} \tag{4.25}$$

$$\begin{aligned}\boldsymbol{e_{j'}} - \boldsymbol{s_i^t}\boldsymbol{e_j} - \boldsymbol{c_i^h} - a + \Delta r_i^t \Delta r_i^{max} - \Delta r_i^t \Delta r_i^{max} + \boldsymbol{s_i^t}\Delta r_i^{max} \\ -\boldsymbol{s_i^t}\Delta r_i^{max} + s - s \leq \boldsymbol{w_i^h}\end{aligned} \tag{4.26}$$

$$\begin{aligned}\boldsymbol{e_{j'}} + \Delta r_i^{max} - (\boldsymbol{s_i^t} + \Delta r_i^t)(\boldsymbol{e_j} + \Delta r_i^{ub}) - (\boldsymbol{c_i^h} - \Delta r_i^t \Delta r_i^{max} \\ -\boldsymbol{s_i^t}\Delta r_i^{max} + \frac{s}{2}) \leq \boldsymbol{w_i^h} + \frac{s}{2}\end{aligned} \tag{4.27}$$

$$\boldsymbol{e_{j'}^*} - \boldsymbol{s_i^{t*}}\boldsymbol{e_j^*} - \boldsymbol{c_i^{h*}} \leq \boldsymbol{w_i^{h*}} \tag{4.28}$$

(>) Let $a := \Delta r_i^{max} - \Delta r_i^t \boldsymbol{e_j} + \Delta r_i^t m(\Delta r_i^t + \boldsymbol{s_i^t})$ and let $s := (\Delta r_i^t + \boldsymbol{s_i^t})\Delta r_i^t m + \Delta r_i^{max}$. Note that a is positive since $\Delta r_i^{max} - \Delta r_i^t \boldsymbol{e_j} > 0$. Therefore, we can perform the following transformations:

$$\boldsymbol{e_{j'}} - \boldsymbol{s_i^t}\boldsymbol{e_j} - \boldsymbol{c_i^h} \geq -\boldsymbol{w_i^h} \tag{4.29}$$

$$\begin{aligned}\boldsymbol{e_{j'}} - \boldsymbol{s_i^t}\boldsymbol{e_j} - \boldsymbol{c_i^h} + a + \Delta r_i^t \Delta r_i^{max} - \Delta r_i^t \Delta r_i^{max} + \boldsymbol{s_i^t}\Delta r_i^{max} \\ -\boldsymbol{s_i^t}\Delta r_i^{max} + \frac{s}{2} - \frac{s}{2} \geq -\boldsymbol{w_i^h}\end{aligned} \tag{4.30}$$

$$\begin{aligned}\boldsymbol{e_{j'}} + \Delta r_i^{max} - (\boldsymbol{s_i^t} + \Delta r_i^t)(\boldsymbol{e_j} + \Delta r_i^{ub}) - (\boldsymbol{c_i^h} - \Delta r_i^t \Delta r_i^{max} \\ -\boldsymbol{s_i^t}\Delta r_i^{max} + \frac{s}{2}) \geq -(\boldsymbol{w_i^h} + \frac{s}{2})\end{aligned} \tag{4.31}$$

$$\boldsymbol{e_{j'}^*} - \boldsymbol{s_i^{t*}}\boldsymbol{e_j^*} - \boldsymbol{c_i^{h*}} \geq -\boldsymbol{w_i^{h*}} \tag{4.32}$$

5. **Case** $i' = i, j' \neq j, k' \neq j, k' = k$:

(<) Let $s := (\Delta r_i^t + \boldsymbol{s_i^t})\Delta r_i^t m + \Delta r_i^{max}$ and let $a := s + \Delta r_i^t \boldsymbol{e_k} - \Delta r_i^{max}$. Note that a is positive since $a = \Delta r_i^t(\boldsymbol{e_k} + m(\Delta r_i^t + \boldsymbol{s_i^t}))$ holds. Therefore, we can perform the following transformations:

$$\boldsymbol{e_{j'}} - \boldsymbol{s_i^t}\boldsymbol{e_k} - \boldsymbol{c_i^h} \leq \boldsymbol{w_i^h} \tag{4.33}$$

$$\begin{aligned}\boldsymbol{e_{j'}} - \boldsymbol{s_i^t}\boldsymbol{e_k} - \boldsymbol{c_i^h} - a + \Delta r_i^t \Delta r_i^{max} - \Delta r_i^t \Delta r_i^{max} + \boldsymbol{s_i^t}\Delta r_i^{max} \\ -\boldsymbol{s_i^t}\Delta r_i^{max} \leq \boldsymbol{w_i^h}\end{aligned} \tag{4.34}$$

$$\begin{aligned}\boldsymbol{e_{j'}} + \Delta r_i^{max} - (\boldsymbol{s_i^t} + \Delta r_i^t)(\boldsymbol{e_k} + \Delta r_i^{max}) - (\boldsymbol{c_i^h} - \Delta r_i^t \Delta r_i^{max} \\ -\boldsymbol{s_i^t}\Delta r_i^{max} + \frac{s}{2}) \leq \boldsymbol{w_i^h} + \frac{s}{2}\end{aligned} \tag{4.35}$$

$$\boldsymbol{e_{j'}^*} - \boldsymbol{s_i^{t*}}\boldsymbol{e_k^*} - \boldsymbol{c_i^{h*}} \leq \boldsymbol{w_i^{h*}} \tag{4.36}$$

(>) Let $s := (\Delta r_i^t + \boldsymbol{s_i^t})\Delta r_i^t m + \Delta r_i^{max}$. Using this definition, we can perform the following transformations:

$$\boldsymbol{e_{j'}} - \boldsymbol{s_i^t}\boldsymbol{e_k} - \boldsymbol{c_i^h} \geq -\boldsymbol{w_i^h} \tag{4.37}$$

$$\begin{aligned}\boldsymbol{e_{j'}} - \boldsymbol{s_i^t}\boldsymbol{e_k} - \boldsymbol{c_i^h} + \Delta r_i^{max} - \Delta r_i^{max} + \Delta r_i^t \Delta r_i^{max} - \Delta r_i^t \Delta r_i^{max} \\ +\boldsymbol{s_i^t}\Delta r_i^{max} - \boldsymbol{s_i^t}\Delta r_i^{max} - \frac{s}{2} \geq -\boldsymbol{w_i^h} - \frac{s}{2}\end{aligned} \tag{4.38}$$

$$\begin{aligned}\boldsymbol{e_{j'}} + \Delta r_i^{max} - (\boldsymbol{s_i^t} + \Delta r_i^t)(\boldsymbol{e_k} + \Delta r_i^{max}) - (\boldsymbol{c_i^h} - \Delta r_i^t \Delta r_i^{max} \\ -\boldsymbol{s_i^t}\Delta r_i^{max} + \frac{s}{2}) \geq -\boldsymbol{w_i^h} - \frac{s}{2}\end{aligned} \tag{4.39}$$

$$\boldsymbol{e_{j'}^*} - \boldsymbol{s_i^{t*}}\boldsymbol{e_k^*} - \boldsymbol{c_i^{h*}} \geq -\boldsymbol{w_i^{h*}} \tag{4.40}$$

6. **Case** $i' = i, j' \neq j, k' \neq j, k' \neq k$:

(<) Let $s := (\Delta r_i^t + \boldsymbol{s_i^t})\Delta r_i^t m + \Delta r_i^{max}$ and let $a := s - \Delta r_i^{max} + \Delta r_i^t \boldsymbol{e_{k'}}$. Note that a is positive since $a = \Delta r_i^t(\boldsymbol{e_{k'}} + m(\Delta r_i^t + \boldsymbol{s_i^t}))$ holds. Therefore, we can perform the following transformations:

$$\boldsymbol{e_{j'}} - \boldsymbol{s_i^t e_{k'}} - \boldsymbol{c_i^h} \leq \boldsymbol{w_i^h} \tag{4.41}$$

$$\begin{aligned}\boldsymbol{e_{j'}} - \boldsymbol{s_i^t e_{k'}} - \boldsymbol{c_i^h} - a + \Delta r_i^t \Delta r_i^{max} - \Delta r_i^t \Delta r_i^{max} + \boldsymbol{s_i^t}\Delta r_i^{max}\\ - \boldsymbol{s_i^t}\Delta r_i^{max} \leq \boldsymbol{w_i^h}\end{aligned} \tag{4.42}$$

$$\begin{aligned}\boldsymbol{e_{j'}} + \Delta r_i^{max} - (\boldsymbol{s_i^t} + \Delta r_i^t)(\boldsymbol{e_{k'}} + \Delta r_i^{max}) - (\boldsymbol{c_i^h} - \Delta r_i^t \Delta r_i^{max}\\ - \boldsymbol{s_i^t}\Delta r_i^{max} + \frac{s}{2}) \leq \boldsymbol{w_i^h} + \frac{s}{2}\end{aligned} \tag{4.43}$$

$$\boldsymbol{e_{j'}^*} - \boldsymbol{s_i^{t*} e_{k'}^*} - \boldsymbol{c_i^{h*}} \leq \boldsymbol{w_i^{h*}} \tag{4.44}$$

(>) Let $a := \Delta r_i^{max} - \Delta r_i^t \boldsymbol{e_{k'}}$ and let $s := (\Delta r_i^t + \boldsymbol{s_i^t})\Delta r_i^t m + \Delta r_i^{max}$. Therefore, we can perform the following transformations:

$$\boldsymbol{e_{j'}} - \boldsymbol{s_i^t e_{k'}} - \boldsymbol{c_i^h} \geq -\boldsymbol{w_i^h} \tag{4.45}$$

$$\begin{aligned}\boldsymbol{e_{j'}} - \boldsymbol{s_i^t e_{k'}} - \boldsymbol{c_i^h} + a + \Delta r_i^t \Delta r_i^{max} - \Delta r_i^t \Delta r_i^{max} + \boldsymbol{s_i^t}\Delta r_i^{max}\\ - \boldsymbol{s_i^t}\Delta r_i^{max} + \frac{s}{2} - \frac{s}{2} \geq -\boldsymbol{w_i^h}\end{aligned} \tag{4.46}$$

$$\begin{aligned}\boldsymbol{e_{j'}} + \Delta r_i^{max} - (\boldsymbol{s_i^t} + \Delta r_i^t)(\boldsymbol{e_{k'}} + \Delta r_i^{max}) - (\boldsymbol{c_i^h} - \Delta r_i^t \Delta r_i^{max}\\ - \boldsymbol{s_i^t}\Delta r_i^{max} + \frac{s}{2}) \geq -(\boldsymbol{w_i^h} + \frac{s}{2})\end{aligned} \tag{4.47}$$

$$\boldsymbol{e_{j'}^*} - \boldsymbol{s_i^{t*} e_{k'}^*} - \boldsymbol{c_i^{h*}} \geq -\boldsymbol{w_i^{h*}} \tag{4.48}$$

7. **Case** $i' \neq i, j' = j, k' \neq j, k' = k$:

(<) Let $s := \boldsymbol{s_{i'}^t}\Delta r_i^t m + \Delta r_i^{max}$ and let $a := s - \Delta r_i^{ub}$. Note that a is positive since $a = \Delta r_i^t m(1 + \boldsymbol{s_{i'}^t}))$ holds. Therefore, we can perform the following transformations:

$$\boldsymbol{e_j} - \boldsymbol{s_{i'}^t e_k} - \boldsymbol{c_{i'}^h} \leq \boldsymbol{w_{i'}^h} \tag{4.49}$$

$$\boldsymbol{e_j} - \boldsymbol{s_{i'}^t e_k} - \boldsymbol{c_{i'}^h} - a + \boldsymbol{s_{i'}^t}\Delta r_i^{max} - \boldsymbol{s_{i'}^t}\Delta r_i^{max} \leq \boldsymbol{w_{i'}^h} \tag{4.50}$$

$$\boldsymbol{e_j} + \Delta r_i^{ub} - \boldsymbol{s_{i'}^t}(\boldsymbol{e_k} + \Delta r_i^{max}) - (\boldsymbol{c_{i'}^h} - \boldsymbol{s_{i'}^t}\Delta r_i^{max} + \frac{s}{2}) \leq \boldsymbol{w_{i'}^h} + \frac{s}{2} \tag{4.51}$$

$$\boldsymbol{e_j^*} - \boldsymbol{s_{i'}^{t*} e_k^*} - \boldsymbol{c_{i'}^{h*}} \leq \boldsymbol{w_{i'}^{h*}} \tag{4.52}$$

(>) Let $a := \Delta r_i^{ub}$ and let $s := \boldsymbol{s^t_{i'}} \Delta r_i^t m + \Delta r_i^{max}$. Note that a is trivially positive since Δr_i^{ub} is positive. Therefore, we can perform the following transformations:

$$\boldsymbol{e_j} - \boldsymbol{s^t_{i'}} \boldsymbol{e_k} - \boldsymbol{c^h_{i'}} \geq -\boldsymbol{w^h_{i'}} \tag{4.53}$$
$$\boldsymbol{e_j} - \boldsymbol{s^t_{i'}} \boldsymbol{e_k} - \boldsymbol{c^h_{i'}} + a + \boldsymbol{s^t_{i'}} \Delta r_i^{max} - \boldsymbol{s^t_{i'}} \Delta r_i^{max} + \frac{s}{2} - \frac{s}{2} \geq -\boldsymbol{w^h_{i'}} \tag{4.54}$$
$$\boldsymbol{e_j} + \Delta r_i^{ub} - \boldsymbol{s^t_{i'}}(\boldsymbol{e_k} + \Delta r_i^{max}) - (\boldsymbol{c^h_{i'}} - \boldsymbol{s^t_{i'}} \Delta r_i^{max} + \frac{s}{2}) \geq -(\boldsymbol{w^h_{i'}} + \frac{s}{2}) \tag{4.55}$$
$$\boldsymbol{e^*_j} - \boldsymbol{s^{t*}_{i'}} \boldsymbol{e^*_k} - \boldsymbol{c^{h*}_{i'}} \geq -\boldsymbol{w^{h*}_{i'}} \tag{4.56}$$

8. **Case** $i' \neq i, j' = j, k' \neq j, k' \neq k$:

   As can be seen easily this case generates the same inequalities as the previous case, except that $k' = k$. Therefore, no relevant difference has to be considered, which is why we omit this case.

9. **Case** $(i' \neq i, j' \neq j, k' = j, k' \neq k)$:

   (<) Let $s := \boldsymbol{s^t_i} \Delta r_i^t m + \Delta r_i^{max}$. Using this definition, we can make the following transformations:

$$\boldsymbol{e_{j'}} - \boldsymbol{s^t_{i'}} \boldsymbol{e_j} - \boldsymbol{c^h_{i'}} \leq \boldsymbol{w^h_{i'}} \tag{4.57}$$
$$\boldsymbol{e_{j'}} - \boldsymbol{s^t_{i'}} \boldsymbol{e_j} - \boldsymbol{c^h_{i'}} + s - s \leq \boldsymbol{w^h_{i'}} \tag{4.58}$$
$$\boldsymbol{e_{j'}} + \Delta r_i^{max} - \boldsymbol{s^t_{i'}}(\boldsymbol{e_j} + \Delta r_i^{ub}) - (\boldsymbol{c^h_{i'}} - \boldsymbol{s^t_{i'}} \Delta r_i^{max} + \frac{s}{2}) \leq \boldsymbol{w^h_{i'}} + \frac{s}{2} \tag{4.59}$$
$$\boldsymbol{e^*_{j'}} - \boldsymbol{s^{t*}_{i'}} \boldsymbol{e^*_j} - \boldsymbol{c^{h*}_{i'}} \leq \boldsymbol{w^{h*}_{i'}} \tag{4.60}$$

   (>) Let $a := \Delta r_i^{max} + \boldsymbol{s^t_i}(\Delta r_i^{max} - \Delta r_i^{ub})$ and let $s := \boldsymbol{s^t_i} \Delta r_i^t m + \Delta r_i^{max}$. Note that a is positive since $\Delta r_i^{max} > \Delta r_i^{ub}$. Therefore, we can perform the following transformations:

$$\boldsymbol{e_{j'}} - \boldsymbol{s^t_{i'}} \boldsymbol{e_j} - \boldsymbol{c^h_{i'}} \geq -\boldsymbol{w^h_{i'}} \tag{4.61}$$
$$\boldsymbol{e_{j'}} - \boldsymbol{s^t_{i'}} \boldsymbol{e_j} - \boldsymbol{c^h_{i'}} + a + \frac{s}{2} - \frac{s}{2} \geq -\boldsymbol{w^h_{i'}} \tag{4.62}$$
$$\boldsymbol{e_{j'}} + \Delta r_i^{max} - \boldsymbol{s^t_{i'}}(\boldsymbol{e_j} + \Delta r_i^{ub}) - (\boldsymbol{c^h_{i'}} - \boldsymbol{s^t_{i'}} \Delta r_i^{max} + \frac{s}{2}) \geq -(\boldsymbol{w^h_{i'}} + \frac{s}{2}) \tag{4.63}$$
$$\boldsymbol{e^*_{j'}} - \boldsymbol{s^{t*}_{i'}} \boldsymbol{e^*_j} - \boldsymbol{c^{h*}_{i'}} \geq -\boldsymbol{w^{h*}_{i'}} \tag{4.64}$$

10. **Case** $i' \neq i, j' \neq j, k' \neq j, k' = k$:

    (<) Let $s := \boldsymbol{s}^t_{i'}\Delta r^t_i m + \Delta r^{max}_i$ and let $a := s - \Delta r^{max}_i$. Note that a is positive since $a = \boldsymbol{s}^t_i \Delta r^t_i m$ holds. Therefore, we can perform the following transformations:

$$\boldsymbol{e_{j'}} - \boldsymbol{s}^t_{i'}\boldsymbol{e_k} - \boldsymbol{c}^h_{i'} \leq \boldsymbol{w}^h_{i'} \tag{4.65}$$
$$\boldsymbol{e_{j'}} - \boldsymbol{s}^t_{i'}\boldsymbol{e_k} - \boldsymbol{c}^h_{i'} - a - \Delta r^{max}_i + \Delta r^{max}_i - \boldsymbol{s}^t_{i'}\Delta r^{max}_i + \boldsymbol{s}^t_{i'}\Delta r^{max}_i \leq \boldsymbol{w}^h_{i'} \tag{4.66}$$
$$\boldsymbol{e_{j'}} + \Delta r^{max}_i - \boldsymbol{s}^t_{i'}(\boldsymbol{e_k} + \Delta r^{max}_i) - (\boldsymbol{c}^h_{i'} - \boldsymbol{s}^t_{i'}\Delta r^{max}_i + \frac{s}{2}) \leq \boldsymbol{w}^h_{i'} + \frac{s}{2} \tag{4.67}$$
$$\boldsymbol{e}^*_{j'} - \boldsymbol{s}^{t*}_{i'}\boldsymbol{e}^*_k - \boldsymbol{c}^{h*}_{i'} \leq \boldsymbol{w}^{h*}_{i'} \tag{4.68}$$

    (>) Let $s := \boldsymbol{s}^t_{i'}\Delta r^t_i m + \Delta r^{max}_i$ and $a := \Delta r^{max}_i$. Note that a is trivially positive since Δr^{max}_i is positive. Therefore, we can perform the following transformations:

$$\boldsymbol{e_{j'}} - \boldsymbol{s}^t_{i'}\boldsymbol{e_k} - \boldsymbol{c}^h_{i'} \geq -\boldsymbol{w}^h_{i'} \tag{4.69}$$
$$\boldsymbol{e_{j'}} - \boldsymbol{s}^t_{i'}\boldsymbol{e_k} - \boldsymbol{c}^h_{i'} + a + \boldsymbol{s}^t_{i'}\Delta r^{max}_i - \boldsymbol{s}^t_{i'}\Delta r^{max}_i + \frac{s}{2} - \frac{s}{2} \geq -\boldsymbol{w}^h_{i'} \tag{4.70}$$
$$\boldsymbol{e_{j'}} + \Delta r^{max}_i - \boldsymbol{s}^t_{i'}(\boldsymbol{e_k} + \Delta r^{max}_i) - (\boldsymbol{c}^h_{i'} - \boldsymbol{s}^t_{i'}\Delta r^{max}_i + \frac{s}{2}) \geq -(\boldsymbol{w}^h_{i'} + \frac{s}{2}) \tag{4.71}$$
$$\boldsymbol{e}^*_{j'} - \boldsymbol{s}^{t*}_{i'}\boldsymbol{e}^*_k - \boldsymbol{c}^{h*}_{i'} \geq -\boldsymbol{w}^{h*}_{i'} \tag{4.72}$$

11. **Case** $i' \neq i, j' \neq j, k' \neq j, k' \neq k$:

    As can be seen easily this case generates the same inequalities as the previous case, except that $k' = k$. Therefore, no relevant difference has to be considered, which is why we omit this case.

12. **Case** $i' \neq i, j' = j, k' = j, k' \neq k$:

    (<) Let $s := \boldsymbol{s}^t_{i'}\Delta r^t_i m + \Delta r^{max}_i$ and let $a := \Delta r^{max}_i - \Delta r^{ub}_i$. Note that a is positive since $a = \Delta r^t_i m$. Therefore, we can perform the following transformations:

$$\boldsymbol{e_j} - \boldsymbol{s}^t_{i'}\boldsymbol{e_j} - \boldsymbol{c}^h_{i'} \leq \boldsymbol{w}^h_{i'} \tag{4.73}$$
$$\boldsymbol{e_j} - \boldsymbol{s}^t_{i'}\boldsymbol{e_j} - \boldsymbol{c}^h_{i'} - a - s + s \leq \boldsymbol{w}^h_{i'} \tag{4.74}$$
$$\boldsymbol{e_j} + \Delta r^{ub}_i - \boldsymbol{s}^t_{i'}(\boldsymbol{e_j} + \Delta r^{ub}_i) - (\boldsymbol{c}^h_{i'} - \boldsymbol{s}^t_{i'}\Delta r^{max}_i + \frac{s}{2}) \leq \boldsymbol{w}^h_{i'} + \frac{s}{2} \tag{4.75}$$
$$\boldsymbol{e}^*_j - \boldsymbol{s}^{t*}_{i'}\boldsymbol{e}^*_j - \boldsymbol{c}^{h*}_{i'} \leq \boldsymbol{w}^{h*}_{i'} \tag{4.76}$$

(>) Let $s := \boldsymbol{s}^{t}_{i'}\Delta r^{t}_{i}m + \Delta r^{max}_{i}$ and $a := \Delta r^{ub}_{i} + \Delta r^{t}_{i}m\boldsymbol{s}^{t}_{i'}$. Note that a is trivially positive since we assumed any parameter to be positive. Therefore, we can perform the following transformations:

$$\boldsymbol{e_j} - \boldsymbol{s}^{t}_{i'}\boldsymbol{e_j} - \boldsymbol{c}^{h}_{i'} \geq -\boldsymbol{w}^{h}_{i'} \tag{4.77}$$

$$\boldsymbol{e_j} - \boldsymbol{s}^{t}_{i'}\boldsymbol{e_j} - \boldsymbol{c}^{h}_{i'} + a + \frac{s}{2} - \frac{s}{2} \geq -\boldsymbol{w}^{h}_{i'} \tag{4.78}$$

$$\boldsymbol{e_j} + \Delta r^{ub}_{i} - \boldsymbol{s}^{t}_{i'}(\boldsymbol{e_j} + \Delta r^{ub}_{i}) - (\boldsymbol{c}^{h}_{i'} - \boldsymbol{s}^{t}_{i'}\Delta r^{max}_{i} + \frac{s}{2}) \geq -(\boldsymbol{w}^{h}_{i'} + \frac{s}{2}) \tag{4.79}$$

$$\boldsymbol{e}^{*}_{j} - \boldsymbol{s}^{t*}_{i'}\boldsymbol{e}^{*}_{j} - \boldsymbol{c}^{h*}_{i'} \geq -\boldsymbol{w}^{h*}_{i'} \tag{4.80}$$

We have shown in any of the twelve discussed cases that if a triple $r_{i'}(e_{j'}, e_{k'})$ with $i' \neq i$ or $j' \neq j$ or $k' \neq k$ was true in the model configuration prior to the induction step, then it is still true in the adjusted model configuration after the induction step. Hence, to show that ExpressivE can capture any self-loop-free graph, it remains to show that any triple that was false remains false after the induction step.

To verify that an initially false tripe $r_{i'}(e_{j'}, e_{k'})$ remains false we solely need to show that the embeddings of $r_{i'}$, $e_{j'}$ and $e_{k'}$ do not satisfy at least one of the Inequalities 4.1 or 4.2. We have to consider the following cases:

1. **Case** $k' \neq k$**:** Any changes to the dimension $\boldsymbol{v}(i,k)$ do not affect the dimension $\boldsymbol{v}(i',k')$. Therefore, if $r_{i'}(e_{j'}, e_{k'})$ for $k' \neq k$ was false before the induction step, it remains false after the induction step, as we solely alter dimension (i,k).

2. **Case** $k' = k, i' = i$**:** In this case $j' \neq j$ needs to hold as the triple $r_i(e_j, e_k)$ was initially assumed to be true. We can easily show that in this case, any triple remains false as follows:

   Let $s := (\Delta r^{t}_{i} + \boldsymbol{s}^{t}_{i})\Delta r^{t}_{i}m + \Delta r^{max}_{i}$, then we can show that our induction step makes $r_i(e_{j'}, e_k)$ false as follows:

$$\boldsymbol{e_{j'}} - \boldsymbol{s}^{t}_{i}\boldsymbol{e_k} - \boldsymbol{c}^{h}_{i} \leq -\boldsymbol{w}^{h}_{i} \tag{4.81}$$

$$\begin{aligned}\boldsymbol{e_{j'}} - \boldsymbol{s}^{t}_{i}\boldsymbol{e_k} - \boldsymbol{c}^{h}_{i} + \Delta r^{max}_{i}(1 - 1 + \Delta r^{t}_{i} - \Delta r^{t}_{i} + \boldsymbol{s}^{t}_{i}\\ -\boldsymbol{s}^{t}_{i}) - \frac{s}{2} \leq -\boldsymbol{w}^{h}_{i} - \frac{s}{2}\end{aligned} \tag{4.82}$$

$$\begin{aligned}\boldsymbol{e_{j'}} + \Delta r^{max}_{i} - (\boldsymbol{s}^{t}_{i} + \Delta r^{t}_{i})(\boldsymbol{e_k} + \Delta r^{max}_{i}) - (\boldsymbol{c}^{h}_{i} - \Delta r^{t}_{i}\Delta r^{max}_{i}\\ -\boldsymbol{s}^{t}_{i}\Delta r^{max}_{i} + \frac{s}{2}) \leq -\boldsymbol{w}^{h}_{i} - \frac{s}{2}\end{aligned} \tag{4.83}$$

$$\boldsymbol{e}^{*}_{j'} - \boldsymbol{s}^{t*}_{i}\boldsymbol{e}^{*}_{k} - \boldsymbol{c}^{h*}_{i} \leq -\boldsymbol{w}^{h*}_{i} \tag{4.84}$$

   Since we started with the complete graph, any triple that is false was made false by an induction step. We have seen that if we apply our algorithm to make $r_i(e_j, e_k)$

false, then Inequality 4.8 holds. Since we assume that $r_i(e_{j'}, e_k)$ was false prior to the current induction step and Inequality 4.8 describes how induction steps make triples false, we can follow that Inequality 4.81 needs to hold prior to this induction step. Next, we add in Inequality 4.82 terms that eliminate each other. Finally, in Inequality 4.83 we restructure the terms such that we can substitute them for the adjusted embedding vectors defined in 1–7. Through this substitution, we obtain Inequality 4.84, which reveals that the adjusted embeddings of $\boldsymbol{e}^*_{\boldsymbol{j'}}$ and $\boldsymbol{e}^*_{\boldsymbol{k}}$ do not lie within the adjusted hyper-parallelogram of relation r_i. Therefore, we have shown that the adjustments of the complete model configuration stated in Steps 1–7 preserve the false triples of this case to remain false.

3. **Case** $i' \neq i$: Any changes to the dimension $\boldsymbol{v}(i, k)$ do not affect the dimension $\boldsymbol{v}(i', k')$. Therefore, if $r_{i'}(e_{j'}, e_{k'})$ for $i' \neq i$ was false before the induction step, it remains false after the induction step, as we solely alter dimension (i', k).

Hence, we have shown that we can make any self-loop-free triple false in the induction step while preserving the truth value of the remaining triples in G. To show fully expressiveness, it remains to show that we can capture any graph G even with self-loops. We started our proof in the base case with a complete graph, which means that any self-loop was initially true. Furthermore, we have shown in Inequalities 4.9–4.16 and 4.73–4.80 that any true self-loop remains true after the induction step and that therefore any constructed complete model configuration captures any self-loop to be true. Since there are only $|R| * |E|$ possibilities to generate triples of the form $r_i(e_j, e_j)$ for any $r_i \in \boldsymbol{R}$ and $e_j \in \boldsymbol{E}$ and since we require just a single dimension where the embedding of the entity pair e_j, e_j is outside of r_i's hyper-parallelogram to make the triple $r_i(e_j, e_j)$ false, we can simply add a dimension per self-loop to our embeddings, whose sole purpose is to exclude one undesired self-loop $r_i(e_j, e_j)$. Therefore, ExpressivE can represent any possible graph G in a complete model configuration of $O(|R| * |E|)$ dimensions, and our model is thus fully expressive in $O(|R| * |E|)$ dimensions. □

4.2.5 Proof of Compositionally Defined Region (Theorem 4.2.3)

This section proves Theorem 4.2.3, which will serve as further machinery for successive sections. First, we extend the notion of when a compositional definition rule *holds* in the virtual triple space ($\mathbb{R}^{2d}$) such that we can employ it later in our proof. Definition 4.2.6 describes when a compositional definition rule holds in dependence of the spatial regions of its relations in $\mathbb{R}^{2d}$. The definition employs the notion of logical implication, i.e., if the body of a rule is satisfied, then its head can be inferred.

Definition 4.2.6 (Truth of Compositional Definition in the Virtual Triple Space)**.** *Let $r_1(X, Y) \wedge r_2(Y, Z) \Leftrightarrow r_d(X, Z)$ be a compositional definition rule over some $r_1, r_2, r_d \in \boldsymbol{R}$ and arbitrary $X, Y, Z \in \boldsymbol{E}$. Also, let $\boldsymbol{f_h}$ be a relation assignment function defined over r_1 and r_2. Moreover, let $\boldsymbol{A_d}$ be the spatial region of r_d in the virtual triple space $\mathbb{R}^{2d}$. The compositional definition rule* holds *for the regions of the relations in $\mathbb{R}^{2d}$, i.e., for $\boldsymbol{f_h}(r_1)$,*

$\boldsymbol{f_h}(r_2)$ and $\boldsymbol{A_d}$, if: ($\Rightarrow$) for any entity assignment function $\boldsymbol{f_e}$ and virtual assignment function $\boldsymbol{f_v}$ over $\boldsymbol{f_e}$ if $\boldsymbol{f_v}(X,Y) \in \boldsymbol{f_h}(r_1)$ and $\boldsymbol{f_v}(Y,Z) \in \boldsymbol{f_h}(r_2)$, then $\boldsymbol{f_v}(X,Z)$ must be within the region $\boldsymbol{A_d}$ of r_d. ($\Leftarrow$) For any entity assignment function $\boldsymbol{f_e}$ and virtual assignment function $\boldsymbol{f_v}$ over $\boldsymbol{f_e}$ if $\boldsymbol{f_v}(X,Z)$ is within the region $\boldsymbol{A_d}$ of r_d, then there exists an entity assignment $\boldsymbol{f_e}(Y)$ such that $\boldsymbol{f_v}(X,Y) \in \boldsymbol{f_h}(r_1)$ and $\boldsymbol{f_v}(Y,Z) \in \boldsymbol{f_h}(r_2)$.

Recall that Theorem 4.2.3 (reformulated in the definitions of Section 4.2.3 and Definition 4.2.6) states that if $\phi := r_1(X,Y) \wedge r_2(Y,Z) \Leftrightarrow r_d(X,Z)$ is a compositional definition rule defined over relations $r_1, r_2, r_d \in \boldsymbol{R}$ and if $\boldsymbol{f_h}$ is a relation assignment function that is defined over r_1 and r_2, then there exists a convex region $\boldsymbol{A_d}$ for r_d in the virtual triple space $\mathbb{R}^{2d}$ such that ϕ holds for $\boldsymbol{f_h}(r_1)$, $\boldsymbol{f_h}(r_2)$, and $\boldsymbol{A_d}$. In particular, we are not only interested in proving the existence of the compositionally defined region $\boldsymbol{A_d}$, but we will even identify a system of inequalities that describes the shape of $\boldsymbol{A_d}$. Specifically, Theorem 4.2.7 concretely characterizes the shape of $\boldsymbol{A_d}$, which we prove subsequently.

Theorem 4.2.7. *Let $r_1(X,Y) \wedge r_2(Y,Z) \Leftrightarrow r_d(X,Z)$ be a compositional definition rule over some relations $r_1, r_2, r_d \in \boldsymbol{R}$ and over arbitrary entities $X, Y, Z \in \boldsymbol{E}$. Furthermore, let $\boldsymbol{f_h}$ be a relation assignment function that is defined over r_1 and r_2 such that for any $i \in \{1,2\}$, $\boldsymbol{f_h}(r_i) = (\boldsymbol{c_i}, \boldsymbol{w_i}, \boldsymbol{s_i})$ with $\boldsymbol{c_i} = (\boldsymbol{c_i^h} || \boldsymbol{c_i^t})$, $\boldsymbol{w_i} = (\boldsymbol{w_i^h} || \boldsymbol{w_i^t})$, and $\boldsymbol{s_i} = (\boldsymbol{s_i^t} || \boldsymbol{s_i^h})$. Moreover, let the slope vectors be positive, i.e., $\boldsymbol{s_i} \succeq \boldsymbol{0}$ for $i \in \{1,2\}$. If Inequalities 4.85–4.90 define the region $\boldsymbol{A_d}$ of r_d in the virtual triple space, then $r_1(X,Y) \wedge r_2(Y,Z) \Leftrightarrow r_d(X,Z)$ holds for $\boldsymbol{f_h}(r_1)$, $\boldsymbol{f_h}(r_2)$ and $\boldsymbol{A_d}$ in the virtual triple space.*

$$(\boldsymbol{x} - \boldsymbol{z}\boldsymbol{s_1^t}\boldsymbol{s_2^t} - \boldsymbol{c_2^h}\boldsymbol{s_1^t} - \boldsymbol{c_1^h})^{|.|} \preceq \boldsymbol{w_2^h}\boldsymbol{s_1^t} + \boldsymbol{w_1^h} \tag{4.85}$$

$$(\boldsymbol{z}\boldsymbol{s_2^t} + \boldsymbol{c_2^h} - \boldsymbol{x}\boldsymbol{s_1^h} - \boldsymbol{c_1^t})^{|.|} \preceq \boldsymbol{w_1^t} + \boldsymbol{w_2^h} \tag{4.86}$$

$$(\boldsymbol{z} - \boldsymbol{x}\boldsymbol{s_1^h}\boldsymbol{s_2^h} - \boldsymbol{c_1^t}\boldsymbol{s_2^h} - \boldsymbol{c_2^t})^{|.|} \preceq \boldsymbol{w_1^t}\boldsymbol{s_2^h} + \boldsymbol{w_2^t} \tag{4.87}$$

$$(\boldsymbol{z} + (\boldsymbol{c_1^h} - \boldsymbol{x})\boldsymbol{s_2^h} \oslash \boldsymbol{s_1^t} - \boldsymbol{c_2^t})^{|.|} \preceq \boldsymbol{w_1^h}\boldsymbol{s_2^h} \oslash \boldsymbol{s_1^t} + \boldsymbol{w_2^t} \tag{4.88}$$

$$(\boldsymbol{x}(\boldsymbol{1} - \boldsymbol{s_1^h}\boldsymbol{s_1^t}) - \boldsymbol{c_1^t}\boldsymbol{s_1^t} - \boldsymbol{c_1^h})^{|.|} \preceq \boldsymbol{w_1^t}\boldsymbol{s_1^t} + \boldsymbol{w_1^h} \tag{4.89}$$

$$(\boldsymbol{z}(\boldsymbol{1} - \boldsymbol{s_2^h}\boldsymbol{s_2^t}) - \boldsymbol{c_2^h}\boldsymbol{s_2^h} - \boldsymbol{c_2^t})^{|.|} \preceq \boldsymbol{w_2^h}\boldsymbol{s_2^h} + \boldsymbol{w_2^t} \tag{4.90}$$

Proof. Let $r_1(X,Y) \wedge r_2(Y,Z) \Leftrightarrow r_d(X,Z)$ be a compositional definition rule over some relations $r_1, r_2, r_d \in \boldsymbol{R}$ and over arbitrary entities $X, Y, Z \in \boldsymbol{E}$. Furthermore, let $\boldsymbol{f_h}$ be a relation assignment function that is defined over r_1 and r_2 such that for any $i \in \{1,2\}$, $\boldsymbol{f_h}(r_i) = (\boldsymbol{c_i}, \boldsymbol{w_i}, \boldsymbol{s_i})$ with $\boldsymbol{c_i} = (\boldsymbol{c_i^h} || \boldsymbol{c_i^t})$, $\boldsymbol{w_i} = (\boldsymbol{w_i^h} || \boldsymbol{w_i^t})$, and $\boldsymbol{s_i} = (\boldsymbol{s_i^t} || \boldsymbol{s_i^h})$. Moreover, let the slope vectors be positive, i.e., $\boldsymbol{s_i} \succeq \boldsymbol{0}$ for $i \in \{1,2\}$.

What we want to show is that if Inequalities 4.85–4.90 define the region of r_d in the virtual triple space, then $r_1(X,Y) \wedge r_2(Y,Z) \Leftrightarrow r_d(X,Z)$ holds in the virtual triple space, i.e., for any entity assignment function $\boldsymbol{f_e}$ and virtual assignment function $\boldsymbol{f_v}$ over $\boldsymbol{f_e}$ if $\boldsymbol{f_v}(X,Y) \in \boldsymbol{f_h}(r_1)$ and $\boldsymbol{f_v}(Y,Z) \in \boldsymbol{f_h}(r_2)$, then $\boldsymbol{f_v}(X,Z)$ must be within the region of r_d. To prove this, we will construct a system of inequalities first that describes r_d and satisfies the compositional definition rule. Afterward, we will show that the constructed system of inequalities has the same behavior as Inequalities 4.85–4.90, proving Theorem 4.2.7.

($\Rightarrow$) First, we choose an arbitrary entity assignment function $\boldsymbol{f_e}$ and virtual assignment function $\boldsymbol{f_v}$ over $\boldsymbol{f_e}$. We will henceforth denote the assigned entity embeddings with $\boldsymbol{f_e}(X) = \boldsymbol{x}$, $\boldsymbol{f_e}(Y) = \boldsymbol{y}$, and $\boldsymbol{f_e}(Z) = \boldsymbol{z}$ to state our proofs concisely. Next, we assume that the left part of $r_1(X,Y) \wedge r_2(Y,Z) \Leftrightarrow r_d(X,Z)$ is true, i.e., that $\boldsymbol{f_v}(X,Y) \in \boldsymbol{f_h}(r_1)$ and $\boldsymbol{f_v}(Y,Z) \in \boldsymbol{f_h}(r_2)$ hold. This means concretely that we can instantiate the following inequalities from Inequalities 4.1–4.2:

$$\boldsymbol{x} - \boldsymbol{c_1^h} - \boldsymbol{s_1^t} \odot \boldsymbol{y} - \boldsymbol{w_1^h} \preceq 0 \tag{4.91}$$
$$\boldsymbol{x} - \boldsymbol{c_1^h} - \boldsymbol{s_1^t} \odot \boldsymbol{y} + \boldsymbol{w_1^h} \succeq 0 \tag{4.92}$$
$$\boldsymbol{y} - \boldsymbol{c_1^t} - \boldsymbol{s_1^h} \odot \boldsymbol{x} - \boldsymbol{w_1^t} \preceq 0 \tag{4.93}$$
$$\boldsymbol{y} - \boldsymbol{c_1^t} - \boldsymbol{s_1^h} \odot \boldsymbol{x} + \boldsymbol{w_1^t} \succeq 0 \tag{4.94}$$

$$\boldsymbol{y} - \boldsymbol{c_2^h} - \boldsymbol{s_2^t} \odot \boldsymbol{z} - \boldsymbol{w_2^h} \preceq 0 \tag{4.95}$$
$$\boldsymbol{y} - \boldsymbol{c_2^h} - \boldsymbol{s_2^t} \odot \boldsymbol{z} + \boldsymbol{w_2^h} \succeq 0 \tag{4.96}$$
$$\boldsymbol{z} - \boldsymbol{c_2^t} - \boldsymbol{s_2^h} \odot \boldsymbol{y} - \boldsymbol{w_2^t} \preceq 0 \tag{4.97}$$
$$\boldsymbol{z} - \boldsymbol{c_2^t} - \boldsymbol{s_2^h} \odot \boldsymbol{y} + \boldsymbol{w_2^t} \succeq 0 \tag{4.98}$$

Our next goal is to construct a system of inequalities that makes $r_d(X,Z)$ — the right part of the rule — true, i.e., that defines the region of r_d such that $\boldsymbol{f_v}(X,Z)$ lies within it. To reach this goal, we substitute Inequalities 4.91–4.98 into each other to receive a system of inequalities that (1) has the same behavior as the initial set and (2) does not contain the entity embedding $\boldsymbol{y}$. Since we have in the beginning assumed that the slope vectors are positive, we can substitute Inequalities 4.91–4.98 into each other as follows:

1. 4.96 in 4.92 and 4.95 in 4.91 leading to 4.99
2. 4.96 in 4.93 and 4.95 in 4.94 leading to 4.100
3. 4.94 in 4.98 and 4.93 in 4.97 leading to 4.101
4. 4.92 in 4.97 and 4.91 in 4.98 leading to 4.102.
5. 4.91 in 4.93 and 4.94 in 4.92 leading to 4.103.
6. 4.95 in 4.97 and 4.98 in 4.96 leading to 4.104.
7. 4.91 in 4.92 leading to 4.105.
8. 4.94 in 4.93 leading to 4.106.
9. 4.95 in 4.96 leading to 4.107.
10. 4.98 in 4.97 leading to 4.108.

These substitutions result in a system of inequalities with the same behavior as the initial system of inequalities. We have listed the result of these substitutions in Inequalities 4.99–4.108.

$$(\boldsymbol{x} - \boldsymbol{z}\boldsymbol{s}_{\mathbf{1}}^{t}\boldsymbol{s}_{\mathbf{2}}^{t} - \boldsymbol{c}_{\mathbf{2}}^{h}\boldsymbol{s}_{\mathbf{1}}^{t} - \boldsymbol{c}_{\mathbf{1}}^{h})^{|.|} \preceq \boldsymbol{w}_{\mathbf{2}}^{h}\boldsymbol{s}_{\mathbf{1}}^{t} + \boldsymbol{w}_{\mathbf{1}}^{h} \tag{4.99}$$

$$(\boldsymbol{z}\boldsymbol{s}_{\mathbf{2}}^{t} + \boldsymbol{c}_{\mathbf{2}}^{h} - \boldsymbol{x}\boldsymbol{s}_{\mathbf{1}}^{h} - \boldsymbol{c}_{\mathbf{1}}^{t})^{|.|} \preceq \boldsymbol{w}_{\mathbf{1}}^{t} + \boldsymbol{w}_{\mathbf{2}}^{h} \tag{4.100}$$

$$(\boldsymbol{z} - \boldsymbol{x}\boldsymbol{s}_{\mathbf{1}}^{h}\boldsymbol{s}_{\mathbf{2}}^{h} - \boldsymbol{c}_{\mathbf{1}}^{t}\boldsymbol{s}_{\mathbf{2}}^{h} - \boldsymbol{c}_{\mathbf{2}}^{t})^{|.|} \preceq \boldsymbol{w}_{\mathbf{1}}^{t}\boldsymbol{s}_{\mathbf{2}}^{h} + \boldsymbol{w}_{\mathbf{2}}^{t} \tag{4.101}$$

$$(\boldsymbol{z} + (\boldsymbol{c}_{\mathbf{1}}^{h} - \boldsymbol{x})\boldsymbol{s}_{\mathbf{2}}^{h} \oslash \boldsymbol{s}_{\mathbf{1}}^{t} - \boldsymbol{c}_{\mathbf{2}}^{t})^{|.|} \preceq \boldsymbol{w}_{\mathbf{1}}^{h}\boldsymbol{s}_{\mathbf{2}}^{h} \oslash \boldsymbol{s}_{\mathbf{1}}^{t} + \boldsymbol{w}_{\mathbf{2}}^{t} \tag{4.102}$$

$$(\boldsymbol{x}(\mathbf{1} - \boldsymbol{s}_{\mathbf{1}}^{h}\boldsymbol{s}_{\mathbf{1}}^{t}) - \boldsymbol{c}_{\mathbf{1}}^{t}\boldsymbol{s}_{\mathbf{1}}^{t} - \boldsymbol{c}_{\mathbf{1}}^{h})^{|.|} \preceq \boldsymbol{w}_{\mathbf{1}}^{t}\boldsymbol{s}_{\mathbf{1}}^{t} + \boldsymbol{w}_{\mathbf{1}}^{h} \tag{4.103}$$

$$(\boldsymbol{z}(\mathbf{1} - \boldsymbol{s}_{\mathbf{2}}^{h}\boldsymbol{s}_{\mathbf{2}}^{t}) - \boldsymbol{c}_{\mathbf{2}}^{h}\boldsymbol{s}_{\mathbf{2}}^{h} - \boldsymbol{c}_{\mathbf{2}}^{t})^{|.|} \preceq \boldsymbol{w}_{\mathbf{2}}^{h}\boldsymbol{s}_{\mathbf{2}}^{h} + \boldsymbol{w}_{\mathbf{2}}^{t} \tag{4.104}$$

$$\boldsymbol{w}_{\mathbf{1}}^{h} \preceq -\boldsymbol{w}_{\mathbf{1}}^{h} \tag{4.105}$$

$$\boldsymbol{w}_{\mathbf{1}}^{t} \preceq -\boldsymbol{w}_{\mathbf{1}}^{t} \tag{4.106}$$

$$\boldsymbol{w}_{\mathbf{2}}^{h} \preceq -\boldsymbol{w}_{\mathbf{2}}^{h} \tag{4.107}$$

$$\boldsymbol{w}_{\mathbf{2}}^{t} \preceq -\boldsymbol{w}_{\mathbf{2}}^{t} \tag{4.108}$$

Note that Inequalities 4.99–4.104 are equivalent to Inequalities 4.85–4.90 and that Inequalities 4.105–4.108 are tautologies since any width embedding $\boldsymbol{w}_{\boldsymbol{i}}^{\boldsymbol{p}}$ is positive by the definition of the ExpressivE model. Therefore, Inequalities 4.99–4.108 and Inequalities 4.85–4.90 have the same behavior, as required. It remains to show that Inequalities 4.99–4.108 define a region $\boldsymbol{A_d}$ containing $\boldsymbol{f_v}(X, Z)$ if $\boldsymbol{f_v}(X, Y) \in \boldsymbol{f_h}(r_1)$ and $\boldsymbol{f_v}(Y, Z) \in \boldsymbol{f_h}(r_2)$. This is trivially true since Inequalities 4.99–4.108 directly follow from Inequalities 4.91–4.98, which are instantiations of Inequalities 4.1–4.2 representing $\boldsymbol{f_v}(X, Y) \in \boldsymbol{f_h}(r_1)$ and $\boldsymbol{f_v}(Y, Z) \in \boldsymbol{f_h}(r_2)$.

Reading the proof bottom-up proves the other direction ($\Leftarrow$), i.e., if $\boldsymbol{f_v}(X, Z)$ is in $\boldsymbol{A_d}$, then there exists an entity assignment $\boldsymbol{f_e}(Y) = \boldsymbol{y}$ such that $\boldsymbol{f_v}(X, Y) \in \boldsymbol{f_h}(r_1)$ and $\boldsymbol{f_v}(Y, Z) \in \boldsymbol{f_h}(r_2)$. Thereby, we have successfully shown that if Inequalities 4.85–4.90 describe the region $\boldsymbol{A_d}$ of relation r_d in the virtual triple space, then $r_1(X, Y) \wedge r_2(Y, Z) \Leftrightarrow r_d(X, Z)$ holds for $\boldsymbol{f_h}(r_1)$, $\boldsymbol{f_h}(r_2)$, and $\boldsymbol{A_d}$ in the virtual triple space. $\square$

We have proven Theorem 4.2.7 in this section, i.e., that Inequalities 4.85–4.90 define the compositionally defined region for positive slope vectors. The proof works vice versa for any other sign of slope vectors, except that the substitutions of Inequalities 4.91–4.98 may vary due to the different signs of slope vectors. Note that by proving Theorem 4.2.7, we have also proven Theorem 4.2.3 — i.e., that there exists a convex region that describes the compositionally defined region $\boldsymbol{A_d}$ — since (1) we have characterized the compositionally defined region and thereby implicitly proven its existence and since (2) Inequalities 4.85–4.90 trivially form a convex region.

4.2.6 Proofs for Exactly Part of Theorems 4.2.2 and 4.2.4

Before we prove the inference capabilities of ExpressivE in this section, we formally define the considered rules in Definition 4.2.8.

Definition 4.2.8. *(Abboud et al., 2020; Pavlović and Sallinger, 2023b) In accordance with Sun et al. (2019); Abboud et al. (2020), we define the following inference rules:*

- *rules of the form* $r_1(X, Y) \Rightarrow r_1(Y, X)$ *with* $r_1 \in \boldsymbol{R}$ *are called* symmetry rules*.*
- *rules of the form* $r_1(X, Y) \wedge r_1(Y, X) \Rightarrow \bot$ *with* $r_1 \in \boldsymbol{R}$ *are called* anti-symmetry rules*.*
- *rules of the form* $r_1(X, Y) \Leftrightarrow r_2(Y, X)$ *with* $r_1, r_2 \in \boldsymbol{R}$ *and* $r_1 \neq r_2$ *are called* inversion rules*.*
- *rules of the form* $r_1(X, Y) \wedge r_2(Y, Z) \Rightarrow r_3(X, Z)$ *with* $r_1, r_2, r_3 \in \boldsymbol{R}$ *and* $r_1 \neq r_2 \neq r_3$ *are called* general composition rules*.*
- *Rules of the form* $r_1(X, Y) \wedge r_2(Y, Z) \Leftrightarrow r_d(X, Z)$ *with* $r_1, r_2, r_d \in \boldsymbol{R}$ *and* $r_1 \neq r_2 \neq r_d$ *are called* compositional definition rules*.*
- *rules of the form* $r_1(X, Y) \Rightarrow r_2(X, Y)$ *with* $r_1, r_2 \in \boldsymbol{R}$ *and* $r_1 \neq r_2$ *are called* hierarchy rules*.*
- *rules of the form* $r_1(X, Y) \wedge r_2(X, Y) \Rightarrow r_3(X, Y)$ *with* $r_1, r_2, r_3 \in \boldsymbol{R}$ *and* $r_1 \neq r_2 \neq r_3$ *are called* intersection rules*.*
- *rules of the form* $r_1(X, Y) \wedge r_2(X, Y) \Rightarrow \bot$ *with* $r_1, r_2 \in \boldsymbol{R}$ *and* $r_1 \neq r_2$ *are called* mutual exclusion rules*.*

With all definitions in place, we prove the exactness part of Theorems 4.2.2 and 4.2.4, i.e., that ExpressivE captures all core inference rules (defined in Chapter 2) exactly. Specifically, we do not solely prove that ExpressivE captures all core inference rules exactly, but that ExpressivE captures these rules exactly iff its relation hyper-parallelograms meet the properties intuitively described in Section 4.2. Next, in Section 4.2.7, we prove that ExpressivE captures rules exactly and exclusively. For the upcoming proofs, we employ the definitions and formal specifications of Sections 4.2.3 and 4.2.5:

Proposition 4.2.9 (Symmetry (Exactly))**.** *Let* $\boldsymbol{m_h} = (\boldsymbol{M}, \boldsymbol{f_h})$ *be a relation configuration and* $r_1 \in \boldsymbol{R}$ *be a symmetric relation, i.e.,* $r_1(X, Y) \Rightarrow r_1(Y, X)$ *holds for any entities* $X, Y \in \boldsymbol{E}$*. Then* $\boldsymbol{m_h}$ *captures* $r_1(X, Y) \Rightarrow r_1(Y, X)$ *exactly iff* r_1*'s relation hyper-parallelogram* $\boldsymbol{f_h(r_1)}$ *is symmetric across the identity line of any correlation subspace.*

Proof. $\Rightarrow$ For the first direction, what is to be shown is that if r_1's relation hyper-parallelogram $\boldsymbol{f_h(r_1)}$ is symmetric across the identity line of any correlation subspace, then $\boldsymbol{m_h}$ captures $r_1(X, Y) \Rightarrow r_1(Y, X)$ exactly. We show this by contradiction. Thus, we first assume that r_1's corresponding relation hyper-parallelogram $\boldsymbol{f_h(r_1)}$ of $\boldsymbol{m_h}$ is symmetric across the identity line for any correlation subspace s_i. Now, to the contrary, we assume that $\boldsymbol{m_h}$ does not capture $r_1(X, Y) \Rightarrow r_1(Y, X)$ exactly. Then, due to the symmetry of the hyper-parallelogram across the identity line in any correlation subspace s_i, for any virtual assignment function $\boldsymbol{f_v}$ it holds that if $\boldsymbol{f_v(e_x, e_y)} \in \boldsymbol{f_h(r_1)}$ for arbitrary entities $e_x, e_y \in \boldsymbol{E}$, then $\boldsymbol{f_v(e_y, e_x)} \in \boldsymbol{f_h(r_1)}$. Yet, by the definition of capturing rules exactly, this means that $\boldsymbol{m_h}$ captures $r_1(X, Y) \Rightarrow r_1(Y, X)$ exactly. This is a contradiction to the initial assumption that $\boldsymbol{m_h}$ does not capture $r_1(X, Y) \Rightarrow r_1(Y, X)$ exactly, proving the $\Rightarrow$ part of the proposition.

$\Leftarrow$ For the second direction, what is to be shown is that if $\boldsymbol{m_h}$ captures $r_1(X, Y) \Rightarrow r_1(Y, X)$ exactly, then r_1's relation hyper-parallelogram $\boldsymbol{f_h(r_1)}$ is symmetric across the identity line of any correlation subspace. We show this by contradiction. Thus, we first assume that $\boldsymbol{m_h}$ captures $r_1(X, Y) \Rightarrow r_1(Y, X)$ exactly, i.e., for any instantiation of $\boldsymbol{f_e}$ and $\boldsymbol{f_v}$ over $\boldsymbol{f_e}$ if $\boldsymbol{f_v(e_x, e_y)} \in \boldsymbol{f_h(r_1)}$, then $\boldsymbol{f_v(e_y, e_x)} \in \boldsymbol{f_h(r_1)}$. Now to the contrary, we assume that r_1's corresponding relation hyper-parallelogram $\boldsymbol{f_h(r_1)}$ of $\boldsymbol{m_h}$ is not symmetric across the identity line in at least one correlation subspace s_i. Then, since $\boldsymbol{f_h(r_1)}$ is not symmetric across the identity line in s_i, there is an instantiation of $\boldsymbol{f_v}$ and $\boldsymbol{f_e}$ such that $\boldsymbol{f_v(e_x, e_y)} \in \boldsymbol{f_h(r_1)}$ and $\boldsymbol{f_v(e_y, e_x)} \notin \boldsymbol{f_h(r_1)}$ for some entities $e_x, e_y \in \boldsymbol{E}$. Yet, by the definition of capturing rules exactly, this means that $\boldsymbol{m_h}$ does not capture $r_1(X, Y) \Rightarrow r_1(Y, X)$ exactly. This is a contradiction to the initial assumption that $\boldsymbol{m_h}$ captures $r_1(X, Y) \Rightarrow r_1(Y, X)$ exactly, proving the $\Leftarrow$ part of the proposition. $\square$

Proposition 4.2.10 (Anti-Symmetry (Exactly))**.** *Let $\boldsymbol{m_h} = (\boldsymbol{M}, \boldsymbol{f_h})$ be a relation configuration and $r_1 \in \boldsymbol{R}$ be an anti-symmetric relation, i.e., $r_1(X, Y) \wedge r_1(Y, X) \Rightarrow \perp$ holds for any entities $X, Y \in \boldsymbol{E}$. Then $\boldsymbol{m_h}$ captures $r_1(X, Y) \wedge r_1(Y, X) \Rightarrow \perp$ exactly iff r_1's relation hyper-parallelogram $\boldsymbol{f_h(r_1)}$ is not symmetric across the identity line in at least one correlation subspace.*

Proposition 4.2.10 can be proven analogously to Proposition 4.2.9. Therefore, its proof has been omitted.

Proposition 4.2.11 (Inversion (Exactly))**.** *Let $\boldsymbol{m_h} = (\boldsymbol{M}, \boldsymbol{f_h})$ be a relation configuration and $r_1, r_2 \in \boldsymbol{R}$ be relations where $r_1(X, Y) \Leftrightarrow r_2(Y, X)$ holds for any entities $X, Y \in \boldsymbol{E}$. Then $\boldsymbol{m_h}$ captures $r_1(X, Y) \Leftrightarrow r_2(Y, X)$ exactly iff $\boldsymbol{f_h(r_1)}$ is the mirror image across the identity line of $\boldsymbol{f_h(r_2)}$ for any correlation subspace.*

Proof. $\Rightarrow$ For the first direction, what is to be shown is that if the relation hyper-parallelogram $\boldsymbol{f_h(r_1)}$ is the mirror image across the identity line of $\boldsymbol{f_h(r_2)}$ for any correlation subspace, then $\boldsymbol{m_h}$ captures $r_1(X, Y) \Leftrightarrow r_2(Y, X)$ exactly. We show this by contradiction. Thus, we first assume that r_1's corresponding relation hyper-parallelogram

$\boldsymbol{f_h(r_1)}$ of $\boldsymbol{m_h}$ is the mirror image across the identity line of $\boldsymbol{f_h(r_2)}$ for any correlation subspace s_i. Now to the contrary, we assume that $\boldsymbol{m_h}$ does not capture $r_1(X,Y) \Leftrightarrow r_2(Y,X)$ exactly. Then, due to $\boldsymbol{f_h(r_1)}$ being the mirror image of $\boldsymbol{f_h(r_2)}$ in any correlation subspace s_i, for any virtual assignment function $\boldsymbol{f_v}$ it holds that if $\boldsymbol{f_v(e_x,e_y)} \in \boldsymbol{f_h(r_1)}$ for arbitrary entities $e_x, e_y \in \boldsymbol{E}$, then $\boldsymbol{f_v(e_y,e_x)} \in \boldsymbol{f_h(r_2)}$. Yet, by the definition of capturing rules exactly, this means that $\boldsymbol{m_h}$ captures $r_1(X,Y) \Leftrightarrow r_2(Y,X)$ exactly. This is a contradiction to the initial assumption that $\boldsymbol{m_h}$ does not capture $r_1(X,Y) \Leftrightarrow r_2(Y,X)$ exactly, proving the $\Rightarrow$ part of the proposition.

$\Leftarrow$ For the second direction, what is to be shown is that if $\boldsymbol{m_h}$ captures $r_1(X,Y) \Leftrightarrow r_2(Y,X)$ exactly, then the relation hyper-parallelogram $\boldsymbol{f_h(r_1)}$ is the mirror image across the identity line of $\boldsymbol{f_h(r_2)}$ for any correlation subspace. We show this by contradiction. Thus, we first assume that $\boldsymbol{m_h}$ captures $r_1(X,Y) \Leftrightarrow r_2(Y,X)$ exactly, i.e., for any instantiation of $\boldsymbol{f_e}$ and $\boldsymbol{f_v}$ over $\boldsymbol{f_e}$ if $\boldsymbol{f_v(e_x,e_y)} \in \boldsymbol{f_h(r_1)}$, then $\boldsymbol{f_v(e_y,e_x)} \in \boldsymbol{f_h(r_2)}$. Now to the contrary, we assume that r_1's corresponding relation hyper-parallelogram $\boldsymbol{f_h(r_1)}$ of $\boldsymbol{m_h}$ is not the mirror image across the identity line of $\boldsymbol{f_h(r_2)}$ for at least one correlation subspace s_i. Then, since $\boldsymbol{f_h(r_1)}$ is not the mirror image across the identity line of $\boldsymbol{f_h(r_2)}$ in s_i, there is an instantiation of $\boldsymbol{f_v}$ and $\boldsymbol{f_e}$ such that $\boldsymbol{f_v(e_x,e_y)} \in \boldsymbol{f_h(r_1)}$ and $\boldsymbol{f_v(e_y,e_x)} \notin \boldsymbol{f_h(r_2)}$ for some entities $e_x, e_y \in \boldsymbol{E}$. Yet, by the definition of capturing rules exactly, this means that $\boldsymbol{m_h}$ does not capture $r_1(X,Y) \Leftrightarrow r_2(Y,X)$ exactly. This is a contradiction to the initial assumption that $\boldsymbol{m_h}$ captures $r_1(X,Y) \Leftrightarrow r_2(Y,X)$ exactly, proving the $\Leftarrow$ part of the proposition. □

Proposition 4.2.12 (Hierarchy (Exactly))**.** *Let* $\boldsymbol{m_h} = (\boldsymbol{M}, \boldsymbol{f_h})$ *be a relation configuration and* $r_1, r_2 \in \boldsymbol{R}$ *be relations where* $r_1(X,Y) \Rightarrow r_2(X,Y)$ *holds for any entities* $X, Y \in \boldsymbol{E}$. *Then* $\boldsymbol{m_h}$ *captures* $r_1(X,Y) \Rightarrow r_2(X,Y)$ *exactly iff* $\boldsymbol{f_h(r_1)}$ *is subsumed by* $\boldsymbol{f_h(r_2)}$ *for any correlation subspace.*

Proof. $\Rightarrow$ For the first direction, what is to be shown is that if the relation hyper-parallelogram $\boldsymbol{f_h(r_1)}$ is subsumed by $\boldsymbol{f_h(r_2)}$ for any correlation subspace, then $\boldsymbol{m_h}$ captures $r_1(X,Y) \Rightarrow r_2(X,Y)$ exactly. We show this by contradiction. Thus, we first assume that r_1's corresponding relation hyper-parallelogram $\boldsymbol{f_h(r_1)}$ of $\boldsymbol{m_h}$ is subsumed by $\boldsymbol{f_h(r_2)}$ for any correlation subspace s_i. Now, to the contrary, we assume that $\boldsymbol{m_h}$ does not capture $r_1(X,Y) \Rightarrow r_2(X,Y)$ exactly. Then, due to $\boldsymbol{f_h(r_1)}$ being a subset of $\boldsymbol{f_h(r_2)}$ in any correlation subspace s_i, for any virtual assignment function $\boldsymbol{f_v}$ it holds that if $\boldsymbol{f_v(e_x,e_y)} \in \boldsymbol{f_h(r_1)}$ for arbitrary entities $e_x, e_y \in \boldsymbol{E}$, then $\boldsymbol{f_v(e_x,e_y)} \in \boldsymbol{f_h(r_2)}$. Yet, by the definition of capturing rules exactly, this means that $\boldsymbol{m_h}$ captures $r_1(X,Y) \Rightarrow r_2(X,Y)$ exactly. This is a contradiction to the initial assumption that $\boldsymbol{m_h}$ does not capture $r_1(X,Y) \Rightarrow r_2(X,Y)$ exactly, proving the $\Rightarrow$ part of the proposition.

$\Leftarrow$ For the second direction, what is to be shown is that if $\boldsymbol{m_h}$ captures $r_1(X,Y) \Rightarrow r_2(X,Y)$ exactly, then the relation hyper-parallelogram $\boldsymbol{f_h(r_1)}$ is subsumed by $\boldsymbol{f_h(r_2)}$ for any correlation subspace. We show this by contradiction. Thus, we first assume that $\boldsymbol{m_h}$ captures $r_1(X,Y) \Rightarrow r_2(X,Y)$ exactly, i.e., for any instantiation of $\boldsymbol{f_e}$ and $\boldsymbol{f_v}$ over $\boldsymbol{f_e}$ if $\boldsymbol{f_v(e_x,e_y)} \in \boldsymbol{f_h(r_1)}$, then $\boldsymbol{f_v(e_x,e_y)} \in \boldsymbol{f_h(r_2)}$. Now to the contrary, we assume that r_1's corresponding relation hyper-parallelogram $\boldsymbol{f_h(r_1)}$ of $\boldsymbol{m_h}$ is not subsumed by $\boldsymbol{f_h(r_2)}$ for at least one correlation subspace s_i. Then, since $\boldsymbol{f_h(r_1)}$ is subsumed by $\boldsymbol{f_h(r_2)}$ in s_i, there is an instantiation of $\boldsymbol{f_v}$ and $\boldsymbol{f_e}$ such that $\boldsymbol{f_v(e_x, e_y)} \in \boldsymbol{f_h(r_1)}$ and $\boldsymbol{f_v(e_x, e_y)} \notin \boldsymbol{f_h(r_2)}$ for some entities $e_x, e_y \in \boldsymbol{E}$. Yet, by the definition of capturing rules exactly, this means that $\boldsymbol{m_h}$ does not capture $r_1(X,Y) \Rightarrow r_2(X,Y)$ exactly. This is a contradiction to the initial assumption that $\boldsymbol{m_h}$ captures $r_1(X,Y) \Rightarrow r_2(X,Y)$ exactly, proving the $\Leftarrow$ part of the proposition. □

Proposition 4.2.13 (Intersection (Exactly))**.** *Let $\boldsymbol{m_h} = (\boldsymbol{M}, \boldsymbol{f_h})$ be a relation configuration and $r_1, r_2, r_3 \in \boldsymbol{R}$ be relations where $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y)$ holds for any entities $X, Y \in \boldsymbol{E}$. Then $\boldsymbol{m_h}$ captures $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y)$ exactly iff the intersection of $\boldsymbol{f_h(r_1)}$ and $\boldsymbol{f_h(r_2)}$ is subsumed by $\boldsymbol{f_h(r_3)}$ for any correlation subspace.*

Proof. $\Rightarrow$ For the first direction, what is to be shown is that if the intersection of $\boldsymbol{f_h(r_1)}$ and $\boldsymbol{f_h(r_2)}$ is subsumed by $\boldsymbol{f_h(r_3)}$ for any correlation subspace, then $\boldsymbol{m_h}$ captures $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y)$ exactly. We show this by contradiction. Thus, we first assume that the intersection of $\boldsymbol{f_h(r_1)}$ and $\boldsymbol{f_h(r_2)}$ of $\boldsymbol{m_h}$ is subsumed by $\boldsymbol{f_h(r_3)}$ for any correlation subspace s_i. Now to the contrary, we assume that $\boldsymbol{m_h}$ does not capture $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y)$ exactly. Then, due to the intersection of $\boldsymbol{f_h(r_1)}$ and $\boldsymbol{f_h(r_2)}$ being a subset of $\boldsymbol{f_h(r_3)}$ in any correlation subspace s_i, for any virtual assignment function $\boldsymbol{f_v}$ it holds that if $\boldsymbol{f_v(e_x, e_y)} \in \boldsymbol{f_h(r_1)}$ and $\boldsymbol{f_v(e_x, e_y)} \in \boldsymbol{f_h(r_2)}$ for arbitrary entities $e_x, e_y \in \boldsymbol{E}$, then $\boldsymbol{f_v(e_x, e_y)} \in \boldsymbol{f_h(r_3)}$. Yet, by the definition of capturing rules exactly, this means that $\boldsymbol{m_h}$ captures $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y)$ exactly. This is a contradiction to the initial assumption that $\boldsymbol{m_h}$ does not capture $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y)$ exactly, proving the $\Rightarrow$ part of the proposition.

$\Leftarrow$ For the second direction, what is to be shown is that if $\boldsymbol{m_h}$ captures $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y)$ exactly, then the intersection of $\boldsymbol{f_h(r_1)}$ and $\boldsymbol{f_h(r_2)}$ is subsumed by $\boldsymbol{f_h(r_3)}$ for any correlation subspace. We show this by contradiction. Thus, we first assume that $\boldsymbol{m_h}$ captures $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y)$ exactly, i.e., for any instantiation of $\boldsymbol{f_e}$ and $\boldsymbol{f_v}$ over $\boldsymbol{f_e}$ if $\boldsymbol{f_v(e_x, e_y)} \in \boldsymbol{f_h(r_1)}$ and $\boldsymbol{f_v(e_x, e_y)} \in \boldsymbol{f_h(r_2)}$, then $\boldsymbol{f_v(e_x, e_y)} \in \boldsymbol{f_h(r_3)}$. Now, to the contrary, we assume that the intersection of $\boldsymbol{f_h(r_1)}$ and $\boldsymbol{f_h(r_2)}$ is not subsumed by $\boldsymbol{f_h(r_3)}$ for at least one correlation subspace s_i. Then, since the intersection of $\boldsymbol{f_h(r_1)}$ and $\boldsymbol{f_h(r_2)}$ is not subsumed by $\boldsymbol{f_h(r_3)}$ in s_i, there is an instantiation of $\boldsymbol{f_v}$ and $\boldsymbol{f_e}$ such that $\boldsymbol{f_v(e_x, e_y)} \in \boldsymbol{f_h(r_1)}$ and $\boldsymbol{f_v(e_x, e_y)} \in \boldsymbol{f_h(r_2)}$ but $\boldsymbol{f_v(e_x, e_y)} \notin \boldsymbol{f_h(r_3)}$ for some entities $e_x, e_y \in \boldsymbol{E}$. Yet, by the definition of capturing rules exactly, this means that $\boldsymbol{m_h}$ does not capture $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y)$ exactly. This is a contradiction to the initial assumption that $\boldsymbol{m_h}$ captures $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y)$ exactly, proving the $\Leftarrow$ part of the proposition. □

Proposition 4.2.14 (Mutual Exclusion (Exactly))**.** *Let $\boldsymbol{m_h} = (\boldsymbol{M}, \boldsymbol{f_h})$ be a relation configuration and $r_1, r_2 \in \boldsymbol{R}$ be relations where $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow \perp$ holds for any entities $X, Y \in \boldsymbol{E}$. Then $\boldsymbol{m_h}$ captures $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow \perp$ exactly iff $\boldsymbol{f_h(r_1)}$ and $\boldsymbol{f_h(r_2)}$ do not intersect in at least one correlation subspace.*

Proof. $\Rightarrow$ For the first direction, what is to be shown is that if the relation hyper-parallelograms $\boldsymbol{f_h(r_1)}$ and $\boldsymbol{f_h(r_2)}$ do not intersect in at least one correlation subspace, then $\boldsymbol{m_h}$ captures $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow \perp$ exactly. We show this by contradiction. Thus, we first assume that $\boldsymbol{f_h(r_1)}$ and $\boldsymbol{f_h(r_2)}$ of $\boldsymbol{m_h}$ do not intersect in at least one correlation subspace s_i. Now, to the contrary, we assume that $\boldsymbol{m_h}$ does not capture $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow \perp$ exactly. Then, since $\boldsymbol{f_h(r_1)}$ and $\boldsymbol{f_h(r_2)}$ do not intersect in at least one correlation subspace s_i, for any virtual assignment function $\boldsymbol{f_v}$ it holds that if $\boldsymbol{f_v(e_x, e_y)} \in \boldsymbol{f_h(r_1)}$ for arbitrary entities $e_x, e_y \in \boldsymbol{E}$, then $\boldsymbol{f_v(e_x, e_y)} \notin \boldsymbol{f_h(r_2)}$. Yet, by the definition of capturing rules exactly, this means that $\boldsymbol{m_h}$ captures $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow \perp$ exactly. This is a contradiction to the initial assumption that $\boldsymbol{m_h}$ does not capture $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow \perp$ exactly, proving the $\Rightarrow$ part of the proposition.

$\Leftarrow$ For the second direction, what is to be shown is that if $\boldsymbol{m_h}$ captures $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow \perp$ exactly, then the relation hyper-parallelograms $\boldsymbol{f_h(r_1)}$ and $\boldsymbol{f_h(r_2)}$ do not intersect in at least one correlation subspace. We show this by contradiction. Thus, we first assume that $\boldsymbol{m_h}$ captures $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow \perp$ exactly, i.e., for any instantiation of $\boldsymbol{f_e}$ and $\boldsymbol{f_v}$ over $\boldsymbol{f_e}$ if $\boldsymbol{f_v(e_x, e_y)} \in \boldsymbol{f_h(r_1)}$, then $\boldsymbol{f_v(e_x, e_y)} \notin \boldsymbol{f_h(r_2)}$ and if $\boldsymbol{f_v(e_x, e_y)} \in \boldsymbol{f_h(r_2)}$, then $\boldsymbol{f_v(e_x, e_y)} \notin \boldsymbol{f_h(r_1)}$. Now to the contrary, we assume that r_1's corresponding relation hyper-parallelogram $\boldsymbol{f_h(r_1)}$ of $\boldsymbol{m_h}$ intersects with $\boldsymbol{f_h(r_2)}$ in any correlation subspace. Then, since $\boldsymbol{f_h(r_1)}$ intersects with $\boldsymbol{f_h(r_2)}$ in any correlation subspace, there is an instantiation of $\boldsymbol{f_v}$ and $\boldsymbol{f_e}$ such that $\boldsymbol{f_v(e_x, e_y)} \in \boldsymbol{f_h(r_1)}$ and $\boldsymbol{f_v(e_x, e_y)} \in \boldsymbol{f_h(r_2)}$ for some entities $e_x, e_y \in \boldsymbol{E}$. Yet, by the definition of capturing rules exactly, this means that $\boldsymbol{m_h}$ does not capture $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow \perp$ exactly. This is a contradiction to the initial assumption that $\boldsymbol{m_h}$ captures $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow \perp$ exactly, proving the $\Leftarrow$ part of the proposition. □

Proposition 4.2.15 (General Composition (Exactly))**.** *Let $r_1, r_2, r_3 \in \boldsymbol{R}$ be relations and let $\boldsymbol{m_h} = (\boldsymbol{M}, \boldsymbol{f_h})$ be a relation configuration, where $\boldsymbol{f_h}$ is defined over r_1, r_2, and r_3. Furthermore, let r_3 be the composite relation of r_1 and r_2, i.e., $r_1(X,Y) \wedge r_2(Y,Z) \Rightarrow r_3(X,Z)$ holds for any entities $X, Y, Z \in \boldsymbol{E}$. Then $\boldsymbol{m_h}$ captures $r_1(X,Y) \wedge r_2(Y,Z) \Rightarrow r_3(X,Z)$ iff the relation hyper-parallelogram $\boldsymbol{f_h(r_3)}$ subsumes the compositionally defined region $\boldsymbol{A_d}$ defined by $\boldsymbol{f_h(r_1)}$ and $\boldsymbol{f_h(r_2)}$ for any correlation subspace.*

Proof. $\Rightarrow$ For the first direction, assume that the compositionally defined region defined by $\boldsymbol{f_h(r_1)}$ and $\boldsymbol{f_h(r_2)}$ is subsumed by $\boldsymbol{f_h(r_3)}$ for any correlation subspace. What is to be shown is that $\boldsymbol{m_h}$ captures $r_1(X,Y) \wedge r_2(Y,Z) \Rightarrow r_3(X,Z)$ exactly. Our proof for this direction is based on the following three results:

1. For an auxiliary relation $r_d \in \boldsymbol{R}$, there exists a convex region $\boldsymbol{A_d}$ in the virtual triple space such that $r_1(X,Y) \wedge r_2(Y,Z) \Leftrightarrow r_d(X,Z)$ holds for $\boldsymbol{f_h(r_1)}$, $\boldsymbol{f_h(r_2)}$, and $\boldsymbol{A_d}$ in any correlation subspace (Theorem 4.2.7).

2. $\boldsymbol{f_h(r_1)}$ subsumes $\boldsymbol{A_d}$ iff $\boldsymbol{m_h}$ captures $r_d(X,Y) \Rightarrow r_3(X,Y)$ exactly (Proposition 4.2.12).

3. $r_1(X, Y) \wedge r_2(Y, Z) \Rightarrow r_3(X, Z)$ logically follows from $\{r_1(X, Y) \wedge r_2(Y, Z) \Leftrightarrow r_d(X, Z),\ r_d(X, Y) \Rightarrow r_3(X, Y)\}$.

For (1), observe that based on Theorem 4.2.7, we know that we can define an auxiliary relation $r_d \in \boldsymbol{R}$ with area $\boldsymbol{A_d}$ such that $r_1(X, Y) \wedge r_2(Y, Z) \Leftrightarrow r_d(X, Z)$ holds for $\boldsymbol{f_h(r_1)}$, $\boldsymbol{f_h(r_2)}$, and $\boldsymbol{A_d}$, i.e., such that $\boldsymbol{A_d}$ is the compositionally defined region of $\boldsymbol{f_h(r_1)}$ and $\boldsymbol{f_h(r_2)}$. For (2), as shown in Proposition 4.2.12, $\boldsymbol{m_h}$ captures $r_d(X, Y) \Rightarrow r_3(X, Y)$ exactly iff $\boldsymbol{f_h(r_3)}$ subsumes r_d's area $\boldsymbol{A_d}$. Therefore, we have shown that if $\boldsymbol{f_h(r_3)}$ subsumes $\boldsymbol{A_d}$, and if $\boldsymbol{A_d}$ is the compositionally defined region of $\boldsymbol{f_h(r_1)}$ and $\boldsymbol{f_h(r_2)}$, then $r_d(X, Y) \Rightarrow r_3(X, Y)$ and $r_1(X, Y) \wedge r_2(Y, Z) \Leftrightarrow r_d(X, Z)$ holds for $\boldsymbol{f_h(r_1)}$, $\boldsymbol{f_h(r_2)}$, $\boldsymbol{f_h(r_3)}$ and $\boldsymbol{A_d}$. Together with the fact that $\boldsymbol{f_h}$ is only defined over r_1, r_2, and r_3, we can infer that $\boldsymbol{m_h}$ exactly captures any rule — solely consisting of r_1, r_2, and r_3 — that follows from $\psi = \{r_1(X, Y) \wedge r_2(Y, Z) \Leftrightarrow r_d(X, Z),\ r_d(X, Y) \Rightarrow r_3(X, Y)\}$. For (3), by logical deduction, the following statement holds: $\psi \models r_1(X, Y) \wedge r_2(Y, Z) \Rightarrow r_3(X, Y)$. Since $r_1(X, Y) \wedge r_2(Y, Z) \Rightarrow r_3(X, Z)$ (i) solely consists of r_1, r_2, and r_3 and (ii) follows from ψ, we have proven that $\boldsymbol{m_h}$ captures $r_1(X, Y) \wedge r_2(Y, Z) \Rightarrow r_3(X, Z)$ exactly if $\boldsymbol{f_h(r_3)}$ subsumes $\boldsymbol{A_d}$, proving the $\Rightarrow$ part of the proposition.

$\Leftarrow$ For the second direction, what is to be shown is that if $\boldsymbol{m_h}$ captures $r_1(X, Y) \wedge r_2(Y, Z) \Rightarrow r_3(X, Z)$ exactly, then the compositionally defined region defined by $\boldsymbol{f_h(r_1)}$ and $\boldsymbol{f_h(r_2)}$ is subsumed by $\boldsymbol{f_h(r_3)}$ for any correlation subspace. We prove this by contradiction. Thus assume that $\boldsymbol{m_h}$ captures $r_1(X, Y) \wedge r_2(Y, Z) \Rightarrow r_3(X, Z)$ exactly, i.e., for any instantiation of $\boldsymbol{f_e}$ and $\boldsymbol{f_v}$ over $\boldsymbol{f_e}$ if $\boldsymbol{f_v(e_x, e_y)} \in \boldsymbol{f_h(r_1)}$ and $\boldsymbol{f_v(e_y, e_z)} \in \boldsymbol{f_h(r_2)}$, then $\boldsymbol{f_v(e_x, e_z)} \in \boldsymbol{f_h(r_3)}$. Now to the contrary, we assume that r_3's corresponding relation hyper-parallelogram $\boldsymbol{f_h(r_3)}$ of $\boldsymbol{m_h}$ does not subsume the compositionally defined region $\boldsymbol{A_d}$ in at least one correlation subspace. The following three points will be used to construct a counter-example: (1) we have shown in Theorem 4.2.7 that we can define an auxiliary relation $r_d \in \boldsymbol{R}$ with area $\boldsymbol{A_d}$ such that $r_1(X, Y) \wedge r_2(Y, Z) \Leftrightarrow r_d(X, Z)$ holds for $\boldsymbol{f_h(r_1)}$, $\boldsymbol{f_h(r_2)}$, and $\boldsymbol{A_d}$, (2) $r_1(X, Y) \wedge r_2(Y, Z) \Rightarrow r_3(X, Z)$ logically follows from $\{r_1(X, Y) \wedge r_2(Y, Z) \Leftrightarrow r_d(X, Z),\ r_d(X, Y) \Rightarrow r_3(X, Y)\}$, stating together with Point (1) and Proposition 4.2.12 that r_3 needs to subsume r_d's area $\boldsymbol{A_d}$ such that $\boldsymbol{m_h}$ can capture $r_1(X, Y) \wedge r_2(Y, Z) \Rightarrow r_3(X, Z)$ exactly, and (3) we have initially assumed that $\boldsymbol{f_h(r_3)}$ does not subsume $\boldsymbol{A_d}$. From (1)-(3) we can infer that there exists an instantiation of $\boldsymbol{f_v}$ and $\boldsymbol{f_e}$ such that $\boldsymbol{f_v(e_x, e_y)} \in \boldsymbol{f_h(r_1)}$ and $\boldsymbol{f_v(e_y, e_z)} \in \boldsymbol{f_h(r_2)}$ but $\boldsymbol{f_v(e_x, e_z)} \notin \boldsymbol{f_h(r_3)}$ for some entities $e_x, e_y, e_z \in \boldsymbol{E}$. Yet, by the definition of capturing rules exactly, this means that $\boldsymbol{m_h}$ does not capture $r_1(X, Y) \wedge r_2(Y, Z) \Rightarrow r_3(X, Z)$ exactly. This is a contradiction to the initial assumption that $\boldsymbol{m_h}$ captures $r_1(X, Y) \wedge r_2(Y, Z) \Rightarrow r_3(X, Z)$ exactly, proving the $\Leftarrow$ part of the proposition. $\square$

Proposition 4.2.16 (Compositional Definition (Exactly))**.** *Let $r_1, r_2, r_d \in \boldsymbol{R}$ be relations and let $\boldsymbol{m_h} = (\boldsymbol{M}, \boldsymbol{f_h})$ be a relation configuration, where $\boldsymbol{f_h}$ is defined over r_1, r_2, and r_d. Also, let r_d be the compositionally defined relation of r_1 and r_2, i.e., $r_1(X, Y) \wedge r_2(Y, Z) \Leftrightarrow r_d(X, Z)$ holds for any entities $X, Y, Z \in \boldsymbol{E}$. Then $\boldsymbol{m_h}$ captures $r_1(X, Y) \wedge r_2(Y, Z) \Leftrightarrow r_d(X, Z)$ iff the relation hyper-parallelogram $\boldsymbol{f_h(r_d)}$ is equal to the compositionally defined region $\boldsymbol{A_d}$ defined by $\boldsymbol{f_h(r_1)}$ and $\boldsymbol{f_h(r_2)}$ for any correlation subspace.*

The proof for Proposition 4.2.16 is straightforward, as Proposition 4.2.16 can be proven analogously to Proposition 4.2.15 with the sole difference that instead of defining a relation embedding $\boldsymbol{f_h}(r_3)$ that subsumes the compositionally defined region $\boldsymbol{A_d}$, we define the compositionally defined relation r_d whose embedding $\boldsymbol{f_h}(r_d)$ is equal to the compositionally defined region $\boldsymbol{A_d}$.

Propositions 4.2.9, 4.2.10, 4.2.11, 4.2.12, 4.2.13, and 4.2.14 together prove the exactness part of Theorem 4.2.2, i.e., that ExpressivE captures symmetry, anti-symmetry, inversion, hierarchy, intersection, and mutual exclusion exactly. Propositions 4.2.15 and 4.2.16 prove the exactness part of Theorem 4.2.4, i.e., that ExpressivE captures general composition and compositional definition exactly. Now it remains to show that ExpressivE captures all these rules exactly and exclusively, which is shown in Section 4.2.7.

4.2.7 Proofs for Exclusively Part of Theorems 4.2.2 and 4.2.4

This section proves that ExpressivE captures all inference rules of Theorems 4.2.2 and 4.2.4 exactly and exclusively. By the definition of capturing a rule ψ exactly and exclusively, this means that we need to construct a relation configuration $\boldsymbol{m_h}$ such that (1) $\boldsymbol{m_h}$ captures ψ and (2) $\boldsymbol{m_h}$ does not capture any positive rule ϕ such that $\psi \not\models \phi$. Note that we have shown in Propositions 4.2.9–4.2.15 that we can construct a relation configuration $\boldsymbol{m_h}$ that captures the following rules by constraining the following geometric properties of $\boldsymbol{m_h}$'s relation hyper-parallelograms:

1. For symmetry and inversion rules, the mirror images across the identity line of hyper-parallelograms in any correlation subspace need to be constrained (Propositions 4.2.9 and 4.2.11).

2. For hierarchy and intersection rules the intersections of hyper-parallelograms in any correlation subspace need to be constrained (Propositions 4.2.12 and 4.2.13).

3. For general composition rules, the compositionally defined region needs to be subsumed in any correlation subspace.

Since symmetry, inversion, hierarchy, intersection, and composition are all positive rules of our considered language of rules, it suffices to analyze the mirror images (M), intersections (I), and compositionally defined regions (C) of each relation hyper-parallelogram to check which positive rules have been captured. Furthermore, for the upcoming proofs, Definition 4.2.17 defines head and tail intervals.

Definition 4.2.17 (Head and Tail Intervals)**.** *Let $r_i \in \boldsymbol{R}$ be a relation and $\boldsymbol{m_h} = (\boldsymbol{M}, \boldsymbol{f_h})$ be a relation configuration. We call an interval a head interval $\boldsymbol{H_{r_i,m_h}}$ and respectively a tail interval $\boldsymbol{T_{r_i,m_h}}$ of r_i and $\boldsymbol{m_h}$ if for arbitrary entities $e_h, e_t \in \boldsymbol{E}$, virtual assignment functions $\boldsymbol{f_v}$, and complete model configuration $\boldsymbol{m}$ over $\boldsymbol{m_h}$ and $\boldsymbol{f_v}$ the following property holds: if $\boldsymbol{m}$ captures a triple $r_1(e_h, e_t)$ to be true, then $\boldsymbol{f_v(e_h)} \in \boldsymbol{H_{r_i,m_h}}$ and $\boldsymbol{f_v(e_t)} \in \boldsymbol{T_{r_i,m_h}}$.*

Using the Definition 4.2.17 and the insights provided by (M), (I), and (C), we will followingly prove that ExpressivE captures each considered rule exactly and exclusively.

Proposition 4.2.18 (Symmetry (Exactly and Exclusively))**.** *Let $\boldsymbol{m_h} = (\boldsymbol{M}, \boldsymbol{f_h})$ be a relation configuration and $r_1 \in \boldsymbol{R}$ be a symmetric relation, i.e., $r_1(X,Y) \Rightarrow r_1(Y,X)$ holds for any entities $X, Y \in \boldsymbol{E}$. Then $\boldsymbol{m_h}$ can capture $r_1(X,Y) \Rightarrow r_1(Y,X)$ exactly and exclusively.*

Proposition 4.2.19 (Anti-Symmetry (Exactly and Exclusively))**.** *Let $\boldsymbol{m_h} = (\boldsymbol{M}, \boldsymbol{f_h})$ be a relation configuration and $r_1 \in \boldsymbol{R}$ be an anti-symmetric relation, i.e., $r_1(X,Y) \wedge r_1(Y,X) \Rightarrow \perp$ holds for any entities $X, Y \in \boldsymbol{E}$. Then $\boldsymbol{m_h}$ can capture $r_1(X,Y) \wedge r_1(Y,X) \Rightarrow \perp$ exactly and exclusively.*

The proofs for Propositions 4.2.18 and 4.2.19 are straightforward, as the only positive rule that contains only one relation is symmetry. Furthermore, since (i) Propositions 4.2.9 and 4.2.10 have shown that there is a relation configuration that can capture symmetry/anti-symmetry exactly and (ii) a hyper-parallelogram cannot be symmetric and anti-symmetric simultaneously, we have shown that there is a relation configuration that captures symmetry/anti-symmetry exactly and exclusively, proving Propositions 4.2.18 and 4.2.19.

Proposition 4.2.20 (Inversion (Exactly and Exclusively))**.** *Let $\boldsymbol{m_h} = (\boldsymbol{M}, \boldsymbol{f_h})$ be a relation configuration and $r_1, r_2 \in \boldsymbol{R}$ be relations where $r_1(X,Y) \Leftrightarrow r_2(Y,X)$ holds for any entities $X, Y \in \boldsymbol{E}$. Then $\boldsymbol{m_h}$ can capture $r_1(X,Y) \Leftrightarrow r_2(Y,X)$ exactly and exclusively.*

The proof for Proposition 4.2.20 is straightforward, as the only positive rules that contain at most two relations are symmetry, hierarchy, and inversion. Furthermore, since (i) Proposition 4.2.11 has shown that there is a relation configuration that can capture inversion exactly and (ii) it is simple to show that a hyper-parallelogram can be the mirror image of another hyper-parallelogram without one of them subsuming the other (hierarchy) or one of them being symmetric across the identity line (symmetry), we have shown that there is a relation configuration that captures inversion exactly and exclusively, proving Proposition 4.2.20.

Proposition 4.2.21 (Hierarchy (Exactly and Exclusively))**.** *Let $\boldsymbol{m_h} = (\boldsymbol{M}, \boldsymbol{f_h})$ be a relation configuration and $r_1, r_2 \in \boldsymbol{R}$ be relations where $r_1(X,Y) \Rightarrow r_2(X,Y)$ holds for any entities $X, Y \in \boldsymbol{E}$. Then $\boldsymbol{m_h}$ can capture $r_1(X,Y) \Rightarrow r_2(X,Y)$ exactly and exclusively.*

The proof for Proposition 4.2.21 is straightforward, as the only positive rules that contain at most two relations are symmetry, hierarchy, and inversion. Furthermore, since (i) Proposition 4.2.12 has shown that there is a relation configuration that can capture hierarchy exactly and (ii) it is simple to show that a hyper-parallelogram can subsume another hyper-parallelogram without one of them being the mirror image across the identity line of the other (inversion) or one of them being symmetric across the identity line (symmetry), we have shown that there is a relation configuration that captures hierarchy exactly and exclusively, proving Proposition 4.2.21.

Proposition 4.2.22 (Intersection (Exactly and Exclusively))**.** *Let $\boldsymbol{m_h} = (\boldsymbol{M}, \boldsymbol{f_h})$ be a relation configuration and $r_1, r_2, r_3 \in \boldsymbol{R}$ be relations where $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y)$ holds for any entities $X, Y \in \boldsymbol{E}$. Then $\boldsymbol{m_h}$ can capture $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y)$ exactly and exclusively.*

Proof. What is to be shown is that $\boldsymbol{m_h}$ can capture intersection ($r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y)$) exactly and exclusively. We have already shown that $\boldsymbol{m_h}$ can capture $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y)$ exactly in Proposition 4.2.13. Now, to show that $\boldsymbol{m_h}$ can capture intersection exactly and exclusively, we construct an instance of $\boldsymbol{m_h}$ such that (1) $\boldsymbol{m_h}$ captures intersection $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y)$ and (2) $\boldsymbol{m_h}$ does not capture any positive rule ϕ such that $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y) \not\models \phi$.

	$\boldsymbol{c^h}$	$\boldsymbol{w^h}$	$\boldsymbol{s^t}$	$\boldsymbol{c^t}$	$\boldsymbol{w^t}$	$\boldsymbol{s^h}$
r_1	-6	2	2	8	2	3
r_2	-11.5	3	5	11	3	3
r_3	-9.5	5	5	9	1	3

Table 4.1: One-dimensional relation embeddings of a relation configuration $\boldsymbol{m_h}$ that captures intersection (i.e., $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y)$) exactly and exclusively.

Figure 4.2 visualizes the hyper-parallelograms defined by the one-dimensional relation embeddings of Table 4.1. In particular, it displays the hyper-parallelograms of r_1, r_2, r_3. As can be easily seen in Figure 4.2 (and proven using Proposition 4.2.13), the relation configuration $\boldsymbol{m_h}$ described by Table 4.1 captures $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y)$ exactly, as $\boldsymbol{f_h(r_3)}$ subsumes the intersection of $\boldsymbol{f_h(r_1)}$ and $\boldsymbol{f_h(r_2)}$.

Now, it remains to show that $\boldsymbol{m_h}$ does not capture any positive rule ϕ such that $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y) \not\models \phi$. To show this, we will show that (M) the mirror image of any relation hyper-parallelogram is not subsumed by any other relation hyper-parallelogram (i.e., no unwanted symmetry nor inversion rule is captured) and (C) the compositionally defined region defined by any pair of hyper-parallelograms is not subsumed by any relation hyper-parallelogram (i.e., no unwanted composition rule is captured). We do not need to show that (I) no unwanted relation hyper-parallelograms intersect, as by the nature of the intersection rule, $\boldsymbol{f_h(r_1)}$, $\boldsymbol{f_h(r_2)}$, and $\boldsymbol{f_h(r_3)}$ should intersect.

For (M), observe in Figure 4.2 that all hyper-parallelograms $\boldsymbol{f_h(r_1)}$, $\boldsymbol{f_h(r_2)}$, and $\boldsymbol{f_h(r_3)}$ of $\boldsymbol{m_h}$ are on the same side of the identity line. Thus, the mirror images of $\boldsymbol{f_h(r_1)}$, $\boldsymbol{f_h(r_2)}$, and $\boldsymbol{f_h(r_3)}$ across the identity line must be on the other side. Therefore, we have shown (M), i.e., that no relation hyper-parallelograms subsume the mirror image of any other relation hyper-parallelogram and thus that $\boldsymbol{m_h}$ does not capture any unwanted symmetry nor inversion rule.

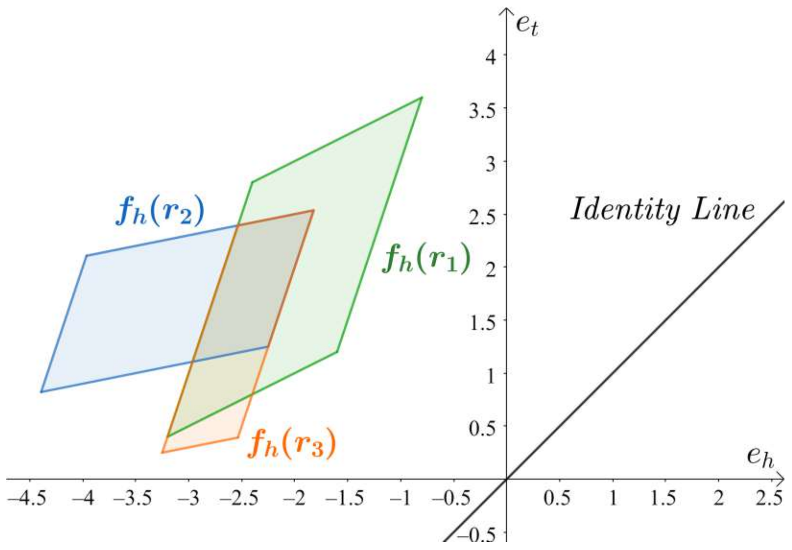

Figure 4.2: Visualization of the relation configuration $\boldsymbol{m_h}$ described by Table 4.1.

For (C), observe in Figure 4.2 that for the displayed relation configuration $\boldsymbol{m_h}$, the head intervals of any relation hyper-parallelogram of $\boldsymbol{m_h}$ contain only negative values and the tail intervals contain only positive values. Thus, for any pair $(r_i, r_j) \in \{r_1, r_2, r_3\}^2$, there is no virtual assignment function $\boldsymbol{f_v}$ such that $\boldsymbol{m}$ over $\boldsymbol{m_h}$ and $\boldsymbol{f_v}$ captures $r_i(x, y)$ and $r_j(y, z)$ for arbitrary entities $x, y, z \in \boldsymbol{E}$. Therefore, no pair of relations (r_i, r_j) defines a compositionally defined region. Thus, we have shown (C) that no compositionally defined region is subsumed by any relation hyper-parallelogram (as no compositionally defined region exists) and that $\boldsymbol{m_h}$ does not capture any unwanted general composition rule.

By Proposition 4.2.13 and by proving (M) and (C), we have shown that the constructed relation configuration $\boldsymbol{m_h}$ of Table 4.1 captures the intersection rule $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y)$ and does not capture any positive rule ϕ such that $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y) \not\models \phi$. This means by the definition of capturing rules exactly and exclusively that $\boldsymbol{m_h}$ captures intersection $(r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y))$ exactly and exclusively, proving the proposition. $\square$

Proposition 4.2.23 (General Composition (Exactly and Exclusively))**.** *Let $r_1, r_2, r_3 \in \boldsymbol{R}$ be relations and let $\boldsymbol{m_h} = (\boldsymbol{M}, \boldsymbol{f_h})$ be a relation configuration, where $\boldsymbol{f_h}$ is defined over r_1, r_2, and r_3. Furthermore, let r_3 be the composite relation of r_1 and r_2, i.e., $r_1(X,Y) \wedge r_2(Y,Z) \Rightarrow r_3(X,Z)$ holds for all entities $X, Y, Z \in \boldsymbol{E}$. Then $\boldsymbol{m_h}$ can capture $r_1(X,Y) \wedge r_2(Y,Z) \Rightarrow r_3(X,Z)$ exactly and exclusively.*

Proof. What is to be shown is that $\boldsymbol{m_h}$ can capture general composition $(r_1(X,Y) \wedge r_2(Y,Z) \Rightarrow r_3(X,Z))$ exactly and exclusively. We have already shown that $\boldsymbol{m_h}$ can capture $r_1(X,Y) \wedge r_2(Y,Z) \Rightarrow r_3(X,Z)$ exactly in Proposition 4.2.15. Now, to show that $\boldsymbol{m_h}$ can capture general composition exactly and exclusively, we construct an instance of

$\boldsymbol{m_h}$ such that (1) $\boldsymbol{m_h}$ captures general composition and (2) $\boldsymbol{m_h}$ does not capture any positive rule ϕ such that $r_1(X,Y) \wedge r_2(Y,Z) \Rightarrow r_3(X,Z) \not\models \phi$.

	$\boldsymbol{c^h}$	$\boldsymbol{w^h}$	$\boldsymbol{s^t}$	$\boldsymbol{c^t}$	$\boldsymbol{w^t}$	$\boldsymbol{s^h}$
r_1	-6	0	2	8	5	3
r_2	-35	5	5	-1	2	5
r_d	-76	10	10	14	2	2.5
r_3	-46	11	6	19	6	4

Table 4.2: One-dimensional relation embeddings of a relation configuration $\boldsymbol{m_h}$ that captures general composition (i.e., $r_1(X,Y) \wedge r_2(Y,Z) \Rightarrow r_3(X,Z)$) and that captures compositional definition (i.e., $r_1(X,Y) \wedge r_2(Y,Z) \Leftrightarrow r_d(X,Z)$) exactly and exclusively.

Figure 4.3 visualizes the hyper-parallelograms defined by the one-dimensional relation embeddings of Table 4.2. In particular, it displays the hyper-parallelograms of r_1, r_2, r_3, and the compositionally defined region $\boldsymbol{A_d}$ of auxiliary relation r_d such that $r_1(X,Y) \wedge r_2(Y,Z) \Leftrightarrow r_d(X,Z)$ holds for $\boldsymbol{f_h(r_1)}$, $\boldsymbol{f_h(r_2)}$, and $\boldsymbol{A_d}$. As can be easily seen in Figure 4.3 (and proven using Theorem 4.2.7 and Proposition 4.2.15), the relation configuration $\boldsymbol{m_h}$ described by Table 4.2 captures $r_1(X,Y) \wedge r_2(Y,Z) \Rightarrow r_3(X,Z)$ exactly, as $\boldsymbol{f_h(r_3)}$ subsumes the compositionally defined region $\boldsymbol{A_d}$.

Now, it remains to show that $\boldsymbol{m_h}$ does not capture any positive rule ϕ such that $r_1(X,Y) \wedge r_2(Y,Z) \Rightarrow r_3(X,Z) \not\models \phi$. To show this, we will show that (M) the mirror image of any relation hyper-parallelogram is not subsumed by any other relation hyper-parallelogram (i.e., no unwanted symmetry nor inversion rule is captured), (I) no relation hyper-parallelograms intersect with each other (i.e., no unwanted hierarchy nor intersection rule is captured), and (C) solely the compositionally defined region $\boldsymbol{A_d}$ defined by $\boldsymbol{f_h(r_1)}$ and $\boldsymbol{f_h(r_2)}$ is subsumed by $\boldsymbol{f_h(r_3)}$ and no other compositionally defined region is subsumed by any other relation hyper-parallelogram (i.e., no unwanted composition rule is captured).

For (M), observe in Figure 4.3 that all hyper-parallelograms $\boldsymbol{f_h(r_1)}$, $\boldsymbol{f_h(r_2)}$, and $\boldsymbol{f_h(r_3)}$ of $\boldsymbol{m_h}$ are on the same side of the identity line. Thus, the mirror images of $\boldsymbol{f_h(r_1)}$, $\boldsymbol{f_h(r_2)}$, and $\boldsymbol{f_h(r_3)}$ across the identity line must be on the other side. Therefore, we have shown (M), i.e., that no relation hyper-parallelograms subsume the mirror image of any other relation hyper-parallelogram and thus that $\boldsymbol{m_h}$ does not capture any unwanted symmetry nor inversion rule.

For (I), observe in Figure 4.3 that no relation hyper-parallelograms $\boldsymbol{f_h(r_1)}$, $\boldsymbol{f_h(r_2)}$, and $\boldsymbol{f_h(r_3)}$ of $\boldsymbol{m_h}$ intersect with each other. Thus, we have shown (I), i.e., that $\boldsymbol{m_h}$ does not capture any unwanted hierarchy nor intersection rule.

For (C), observe in Figure 4.3 that for the displayed relation configuration $\boldsymbol{m_h}$, the following head and tail intervals can be defined: (i) $\boldsymbol{H_{r_1,m_h}} = [-4,0]$ and $\boldsymbol{T_{r_1,m_h}} = [1,3]$, (ii) $\boldsymbol{H_{r_2,m_h}} = [1,3]$ and $\boldsymbol{T_{r_2,m_h}} = [6,9]$, and (iii) $\boldsymbol{H_{r_3,m_h}} = [-6,-1]$ and $\boldsymbol{T_{r_3,m_h}} = [4,10]$. The tail intervals solely overlap with the head intervals for $\boldsymbol{T_{r_1,m_h}}$ and $\boldsymbol{H_{r_2,m_h}}$, i.e.,

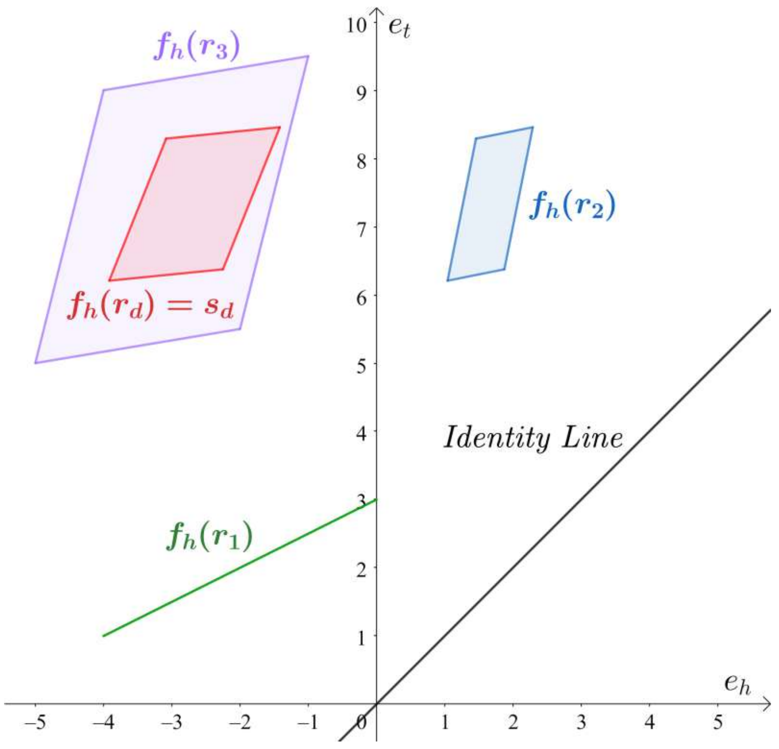

Figure 4.3: Visualization of the relation configuration $\boldsymbol{m_h}$ described by Table 4.2.

$\boldsymbol{T_{r_i,m_h}} \cap \boldsymbol{H_{r_j,m_h}} = \emptyset, (r_i, r_j) \in \{r_1, r_2, r_3\}^2 \setminus (r_1, r_2)$. Thus, for any pair $(r_i, r_j) \in \{r_1, r_2, r_3\}^2 \setminus (r_1, r_2)$ there is no virtual assignment function $\boldsymbol{f_v}$ such that $\boldsymbol{m}$ over $\boldsymbol{m_h}$ and $\boldsymbol{f_v}$ captures $r_i(x, y)$ and $r_j(y, z)$ for arbitrary entities $x, y, z \in \boldsymbol{E}$. Therefore, (r_1, r_2) is the only pair of relations that defines a compositionally defined region, i.e., no other pair of relations defines a compositionally defined region. Thus, we have shown (C) that no other compositionally defined region is subsumed by any other relation (as no other compositionally defined region exists) and thus that no unwanted composition rule is captured by $\boldsymbol{m_h}$.

By Proposition 4.2.15 and by proving (I), (M), and (C), we have shown that the constructed relation configuration $\boldsymbol{m_h}$ of Table 4.2 captures the general composition rule $r_1(X, Y) \wedge r_2(Y, Z) \Rightarrow r_3(X, Z)$ and does not capture any positive rule ϕ such that $r_1(X, Y) \wedge r_2(Y, Z) \Rightarrow r_3(X, Z) \not\models \phi$. This means by the definition of capturing rules exactly and exclusively that $\boldsymbol{m_h}$ captures general composition $(r_1(X, Y) \wedge r_2(Y, Z) \Rightarrow r_3(X, Z))$ exactly and exclusively, proving the proposition. □

Proposition 4.2.24 (Compositional Definition (Exactly and Exclusively))**.** *Let $r_1, r_2, r_d \in \boldsymbol{R}$ be relations and let $\boldsymbol{m_h} = (\boldsymbol{M}, \boldsymbol{f_h})$ be a relation configuration, where $\boldsymbol{f_h}$ is defined over r_1, r_2, and r_d. Furthermore, let r_d be the compositionally defined relation of r_1 and r_2, i.e., $r_1(X, Y) \wedge r_2(Y, Z) \Leftrightarrow r_d(X, Z)$ holds for all entities $X, Y, Z \in \boldsymbol{E}$. Then $\boldsymbol{m_h}$ can capture $r_1(X, Y) \wedge r_2(Y, Z) \Leftrightarrow r_d(X, Z)$ exactly and exclusively.*

The proof for Proposition 4.2.24 is straightforward, as it can be proven analogously to Proposition 4.2.23 with the only difference that instead of defining a relation embedding $\boldsymbol{f_h}(r_3)$ that subsumes the compositionally defined region, we define the compositionally defined relation r_d whose embedding $\boldsymbol{f_h}(r_d)$ is equal to the compositionally defined region $\boldsymbol{A_d}$. We have stated the relation embeddings for r_d in Table 4.2 and also visualized $\boldsymbol{f_h}(r_d)$ in Figure 4.3.

Finally, the sum of Propositions 4.2.18–4.2.24 proves Theorems 4.2.2 and 4.2.4. Thus, we have theoretically shown that ExpressivE captures all core inference rules exactly and exclusively, directly proving Corollary 4.2.5.

4.2.8 Extended Compositions

This section provides theoretical evidence that ExpressivE is not limited to capturing a single composition rule. Specifically, we prove that ExpressivE can capture more than one application of a composition rule. The following theoretical result is empirically backed up by further experimental results of Section 4.5.6.

Proposition 4.2.25. *Let $r_1, r_2, r_3, r_{1,2}, r_{1,2,3} \in \boldsymbol{R}$ be relations and let $\boldsymbol{m_h} = (\boldsymbol{M}, \boldsymbol{f_h})$ be a relation configuration, where $\boldsymbol{f_h}$ is defined over $r_1, r_2, r_3, r_{1,2}$, and $r_{1,2,3}$. Furthermore, let $r_1(X,Y) \land r_2(Y,Z) \Rightarrow r_{1,2}(X,Z)$ and $r_{1,2}(X,Y) \land r_3(Y,Z) \Rightarrow r_{1,2,3}(X,Z)$ hold for all entities $X, Y, Z \in \boldsymbol{E}$. Then $\boldsymbol{m_h}$ can capture $r_1(X,Y) \land r_2(Y,Z) \Rightarrow r_{1,2}(X,Z)$ and $r_{1,2}(X,Y) \land r_3(Y,Z) \Rightarrow r_{1,2,3}(X,Z)$ exactly and exclusively.*

Proof. What is to be shown is that $\boldsymbol{m_h}$ can capture $\phi_1 := r_1(X,Y) \land r_2(Y,Z) \Rightarrow r_{1,2}(X,Z)$ and $\phi_2 := r_{1,2}(X,Y) \land r_3(Y,Z) \Rightarrow r_{1,2,3}(X,Z)$ exactly and exclusively. To show that there is an $\boldsymbol{m_h}$ that captures ϕ_1 and ϕ_2 exactly and exclusively, we construct an instance of $\boldsymbol{m_h}$ such that (1) $\boldsymbol{m_h}$ captures ϕ_1 and ϕ_2 exactly, and (2) $\boldsymbol{m_h}$ does not capture any positive rule ψ such that $(\phi_1 \land \phi_2) \not\models \psi$.

Figure 4.4 visualizes the hyper-parallelograms defined by the one-dimensional relation embeddings of Table 4.3. In particular, it displays the hyper-parallelograms of r_1, r_2, $r_{1,2}$, r_3, $r_{1,2,3}$, and the compositionally defined regions $\boldsymbol{A^d_{1,2}}$, $\boldsymbol{A^d_{2,3}}$, $\boldsymbol{A^d_{(1,2),3}}$, $\boldsymbol{A^d_{1,(2,3)}}$ of auxiliary relation $r^d_{1,2}$, $r^d_{2,3}$ $r^d_{(1,2),3}$, and $r^d_{1,(2,3)}$ such that $r_1(X,Y) \land r_2(Y,Z) \Leftrightarrow r^d_{1,2}(X,Z)$, $r_2(X,Y) \land r_3(Y,Z) \Leftrightarrow r^d_{2,3}(X,Z)$, $r_{1,2}(X,Y) \land r_3(Y,Z) \Leftrightarrow r^d_{(1,2),3}(X,Z)$, and $r_1(X,Y) \land r^d_{2,3}(Y,Z) \Leftrightarrow r^d_{1,(2,3)}(X,Z)$ hold for $\boldsymbol{f_h(r_1)}$, $\boldsymbol{f_h(r_2)}$, $\boldsymbol{f_h(r_3)}$, $\boldsymbol{f_h(r_{1,2})}$, $\boldsymbol{f_h(r_{1,2,3})}$, $\boldsymbol{A^d_{1,2}}$, $\boldsymbol{A^d_{2,3}}$, $\boldsymbol{A^d_{(1,2),3}}$, and $\boldsymbol{A^d_{1,(2,3)}}$. Note that from ϕ_1 and ϕ_2 together with the auxiliary relation $r^d_{1,(2,3)}$ — defined above — follows that $r^d_{1,2}(X,Y) \Rightarrow r_{1,2}(X,Y)$, $r^d_{(1,2),3}(X,Y) \Rightarrow r_{1,2,3}(X,Y)$, and $r^d_{1,(2,3)}(X,Y) \Rightarrow r_{1,2,3}(X,Y)$ need to be satisfied. Thus, as can be easily seen in Figure 4.4 (and proven using Theorem 4.2.7 and Proposition 4.2.15), the relation configuration $\boldsymbol{m_h}$ described by Table 4.3 captures ϕ_1 and ϕ_2 exactly, as $\boldsymbol{f_h(r_{1,2})}$ subsumes the compositionally defined region $\boldsymbol{A^d_{1,2}}$ and as $\boldsymbol{f_h(r_{1,2,3})}$ subsumes the compositionally defined regions $\boldsymbol{A^d_{(1,2),3}}$ and $\boldsymbol{A^d_{1,(2,3)}}$.

	$\boldsymbol{c^h}$	$\boldsymbol{w^h}$	$\boldsymbol{s^t}$	$\boldsymbol{c^t}$	$\boldsymbol{w^t}$	$\boldsymbol{s^h}$
r_1	-6	0	2	8	5	3
r_2	-35	5	5	-1	2	5
$r_{1,2}$	-46	11	6	19	6	4
r_3	-45	3	5	-20	0	4
$r_{1,2,3}$	-215	20	20	22	8	4

Table 4.3: One-dimensional relation embeddings of a relation configuration $\boldsymbol{m_h}$ that captures two general compositions (i.e., $r_1(X,Y) \land r_2(Y,Z) \Rightarrow r_{1,2}(X,Z)$ and $r_{1,2}(X,Y) \land r_3(Y,Z) \Rightarrow r_{1,2,3}(X,Z)$) exactly and exclusively.

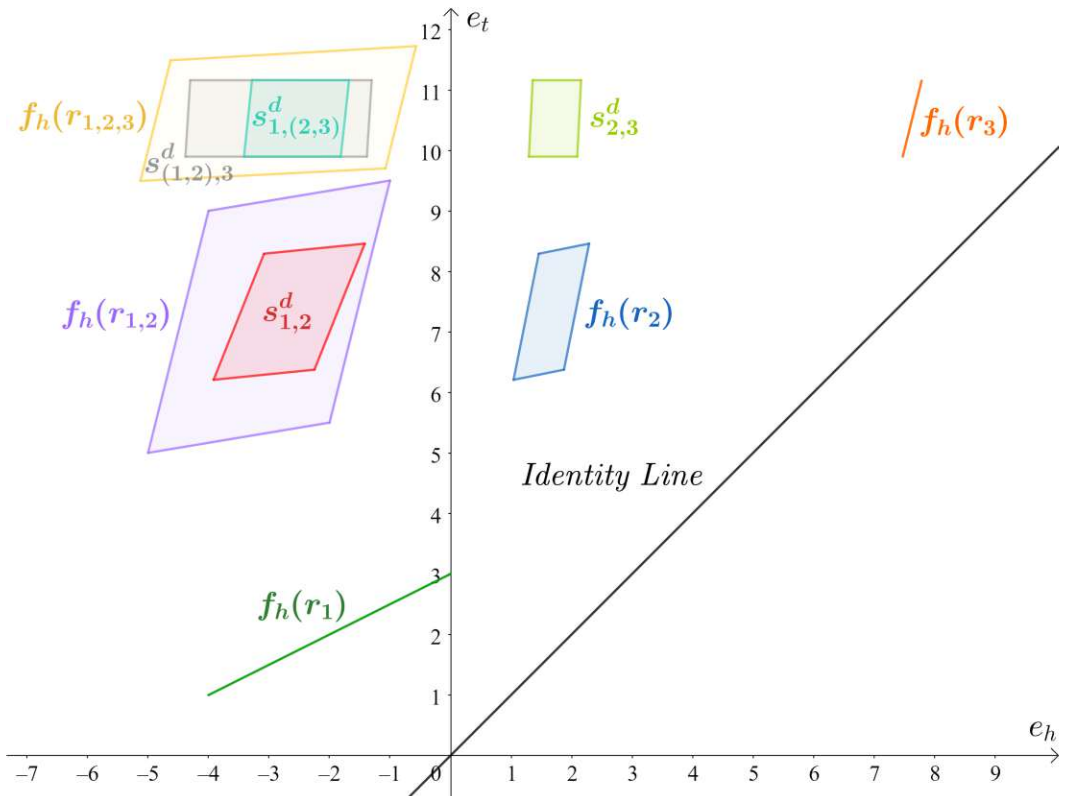

Figure 4.4: Visualization of the relation configuration $\boldsymbol{m_h}$ described by Table 4.3.

Now it remains to show that $\boldsymbol{m_h}$ does not capture any positive rule ψ such that $(\phi_1 \land \phi_2) \not\models \psi$. To show this, we will show that (M) the mirror image of any relation hyper-parallelogram is not subsumed by any other relation hyper-parallelogram (i.e., no unwanted symmetry nor inversion rule is captured), (I) no relation hyper-parallelograms intersect with each other (i.e., no unwanted hierarchy nor intersection rule is captured), and (C) solely that $\boldsymbol{A^d_{1,2}} \subseteq \boldsymbol{f_h(r_{1,2})}$ and $(\boldsymbol{A^d_{(1,2),3}} \cup \boldsymbol{A^d_{1,(2,3)}}) \subseteq \boldsymbol{f_h(r_{1,2,3})}$ are satisfied, and no other compositionally defined region is subsumed by any other relation hyper-parallelogram (i.e., no unwanted composition rule is captured).

For (M), observe in Figure 4.4 that all hyper-parallelograms $\boldsymbol{f_h(r_1)}$, $\boldsymbol{f_h(r_2)}$, $\boldsymbol{f_h(r_3)}$, $\boldsymbol{f_h(r_{1,2})}$, and $\boldsymbol{f_h(r_{1,2,3})}$ of $\boldsymbol{m_h}$ are on the same side of the identity line. Thus, the mirror images of any of these hyper-parallelograms across the identity line must be on the other side. Therefore, we have shown (M), i.e., that no relation hyper-parallelograms subsume the mirror image of any other relation hyper-parallelogram and thus that $\boldsymbol{m_h}$ does not capture any unwanted symmetry nor inversion rule.

For (I), observe in Figure 4.4 that no relation hyper-parallelograms $\boldsymbol{f_h(r_1)}$, $\boldsymbol{f_h(r_2)}$, $\boldsymbol{f_h(r_3)}$, $\boldsymbol{f_h(r_{1,2})}$, and $\boldsymbol{f_h(r_{1,2,3})}$ of $\boldsymbol{m_h}$ intersect with each other. Thus, we have shown (I), i.e., that $\boldsymbol{m_h}$ does not capture any unwanted hierarchy nor intersection rule.

For (C), recall Definition 4.2.17, describing head and tail intervals. We observe in Figure 4.4 that for the displayed relation configuration $\boldsymbol{m_h}$, the following head and tail intervals can be defined: (i) $\boldsymbol{H_{r_1,m_h}} = [-4, 0]$ and $\boldsymbol{T_{r_1,m_h}} = [1, 3]$, (ii) $\boldsymbol{H_{r_2,m_h}} = [1, 3]$ and $\boldsymbol{T_{r_2,m_h}} = [6, 9]$, (iii) $\boldsymbol{H_{r_{1,2},m_h}} = [-6, -1]$ and $\boldsymbol{T_{r_{1,2},m_h}} = [4, 9.7]$, (iv) $\boldsymbol{H_{r_3,m_h}} = [7, 9]$ and $\boldsymbol{T_{r_3,m_h}} = [10, 12]$, (v) $\boldsymbol{H_{r_{1,2,3},m_h}} = [-6, 0]$ and $\boldsymbol{T_{r_{1,2,3},m_h}} = [9.8, 12]$, and (vi) $\boldsymbol{H_{r^d_{2,3},m_h}} = [1, 3]$ and $\boldsymbol{T_{r^d_{2,3},m_h}} = [9.8, 12]$. The tail intervals solely overlap with the head intervals for the pairs $\{(r_1, r_2), (r_2, r_3), (r_{1,2}, r_3), (r_1, r^d_{2,3})\}$, i.e., $\boldsymbol{T_{r_i,m_h}} \cap \boldsymbol{H_{r_j,m_h}} = \emptyset, (r_i, r_j) \in \{r_1, r_2, r_3\}^2 \backslash \{(r_1, r_2), (r_2, r_3), (r_{1,2}, r_3), (r_1, r^d_{2,3})\}$. Thus, for any pair $(r_i, r_j) \in \{r_1, r_2, r_3\}^2 \setminus \{(r_1, r_2), (r_2, r_3), (r_{1,2}, r_3), (r_1, r^d_{2,3})\}$ there is no virtual assignment function $\boldsymbol{f_v}$ such that $\boldsymbol{m}$ over $\boldsymbol{m_h}$ and $\boldsymbol{f_v}$ captures $r_i(x, y)$ and $r_j(y, z)$ for arbitrary entities $x, y, z \in \boldsymbol{E}$. Therefore, $\{(r_1, r_2), (r_2, r_3), (r_{1,2}, r_3), (r_1, r^d_{2,3})\}$ are the only pairs of relations that define a compositionally defined region, i.e., no other pair of relations defines a compositionally defined region. Thus, we have shown that (1) $\boldsymbol{m_h}$ captures ϕ_1 and ϕ_2 exactly — since $\boldsymbol{A^d_{1,2}} \subseteq \boldsymbol{f_h(r_{1,2})}$ and $(\boldsymbol{A^d_{(1,2),3}} \cup \boldsymbol{A^d_{1,(2,3)}}) \subseteq \boldsymbol{f_h(r_{1,2,3})}$ — and (2) the only other existing compositionally defined region $\boldsymbol{A^d_{2,3}}$ is disjoint with any other relation hyper-parallelograms. By (1) and (2), we have shown (C) that no other compositionally defined region (specifically $\boldsymbol{A^d_{1,2}}$) is subsumed by any other relation and thus that no unwanted composition rule is captured by $\boldsymbol{m_h}$.

By proving that the constructed $\boldsymbol{m_h}$ captures ϕ_1 and ϕ_2 exactly and by (I), (M), and (C), we have shown that the constructed relation configuration $\boldsymbol{m_h}$ of Table 4.3 captures ϕ_1 and ϕ_2 and does not capture any positive rule ψ such that $(\phi_1 \wedge \phi_2) \not\models \psi$. This means by the definition of capturing rules exactly and exclusively that $\boldsymbol{m_h}$ captures ϕ_1 and ϕ_2 exactly and exclusively, proving the proposition. □

4.3 ExpressivE's Two Natures

In this section, we analyze functional and spatial models in more detail and outline how ExpressivE combines the capabilities of both model families. ExpressivE has two natures, specifically:

- ExpressivE has a functional nature (in the spirit of functional models such as TransE and RotatE), allowing it to capture functional composition, discussed in detail in Section 4.3.1.
- ExpressivE has a spatial nature (in the spirit of spatial models such as BoxE), allowing it to capture hierarchy, discussed in detail in Section 4.3.2.

The combination of the functional and spatial nature is precisely the reason that allows ExpressivE to capture hierarchy and composition rules jointly. In the following, we review the inference capabilities of spatial and functional models and discuss how ExpressivE combines both the spatial and functional nature.

4.3.1 Analysis of Functional Models

We recall the definition of *functional models* provided in Section 3.1, which states that functional models basically embed relations as functions $\boldsymbol{f_{r_i}} : \mathbb{K}^d \to \mathbb{K}^d$ and entities as vectors $\boldsymbol{e_j} \in \mathbb{K}^d$ over some field $\mathbb{K}$. These models represent true triples $r_i(e_h, e_t)$ as $\boldsymbol{e_t} = \boldsymbol{f_{r_i}}(\boldsymbol{e_h})$ in the embedding space.

Our analysis has revealed that the root cause that functional models cannot capture general composition rules lies within the functional nature of these models. In essence, these models mainly use functions to embed relations. This allows them to leverage functional composition $\boldsymbol{f_{r_d}} = \boldsymbol{f_{r_2}} \circ \boldsymbol{f_{r_1}}$ to capture composition rules. Yet, employing functional composition *defines* the composite relation r_d completely and thus represents a more restricted rule that we call *compositional definition* $r_1(X,Y) \land r_2(Y,Z) \Leftrightarrow r_d(X,Z)$.

In contrast, general composition $r_1(X,Y) \land r_2(Y,Z) \Rightarrow r_3(X,Z)$ does *not* completely define its composite relation r_3. This means that in the case of general composition, the composite relation r_3 may contain more triples than those that are directly *inferable* by compositional definition rules. Due to this notion of extensibility, we can describe general composition as a combination of compositional definition and hierarchy, i.e., a general composition rule defines its composite relation r_3 as a superset (hierarchy component) of the compositionally defined relation r_d. This explains why no KGE has managed to capture general composition, as any state-of-the-art KGE that supports some notion of composition cannot represent hierarchy and vice versa (as will be discussed in Section 4.3.2), yet both are essential to support general composition. Therefore, to capture general composition, ExpressivE combines hierarchy and compositional definition rules, as discussed in more detail in Section 4.2.2.

4.3.2 Analysis of Spatial Models

Spatial models embed a relation $r \in \boldsymbol{R}$ via spatial regions in the embedding space. Furthermore, they embed an entity $e_a \in \boldsymbol{E}$ in the role of a head and tail entity with two independent embeddings $\boldsymbol{e_a^h} \in \mathbb{K}^d$ and $\boldsymbol{e_a^t} \in \mathbb{K}^d$. A triple $r(e_h, e_t)$ is true for spatial models if the embeddings of the entities e_h and e_t lie within the respective spatial regions of the relation r. Thus, spatial models may capture hierarchy rules via the spatial subsumption of the regions defined by the relations. However, since there is no relation between $\boldsymbol{e}_a^h$ and $\boldsymbol{e}_a^t$, spatial models — such as BoxE (Abboud et al., 2020) — cannot capture composition.

ExpressivE embeds relations as regions (spatial nature). Yet, to achieve the functional nature, it cannot use two independent entity embeddings in the typical embedding space - as we discussed above. The solution and key difference to BoxE is to define the virtual triple space, which is formed by concatenating head and tail entity embeddings of the same embedding space (as described in detail in Section 4.1). More specifically, any line through the virtual triple space defines a function between head and tail entity embeddings of the same space - the key to the functional nature:

- **Functional nature.** Regions in this virtual triple space establish a mathematical relation between head and tail entities of the same space by which composition can be captured.
- **Spatial nature.** At the same time, regions can subsume each other, by which - as is intuitive - hierarchy rules can be captured.

Finally, it is precisely the combination of the functional and spatial nature that allows ExpressivE to capture general composition, as described in detail in Section 4.2.2.

4.4 Experimental Setup Details

Before we can discuss empirical results, we need to first introduce our experimental setup. Specifically, this section presents our concrete experimental setup, including details of our implementation, used hardware, learning setup, hyperparameters, and CO_2 emissions.

Implementation Details. We have implemented ExpressivE in PyKEEN 1.7 (Ali et al., 2021), which is a Python library that uses the MIT license and supports many benchmark KGs and KGEs. Thereby, we make ExpressivE comfortably accessible to the community for future benchmarks and experiments. We have made our code publicly available in a GitHub repository[1]. It contains, in addition to the code of ExpressivE, a setup file to install the necessary libraries and a ReadMe.md file containing library versions and running instructions to facilitate the reproducibility of our results.

Ablation Versions. To study the effects of ExpressivE's parameters on its performance, we consider the following constrained ExpressivE versions: (1) *Base ExpressivE*, which represents ExpressivE without any parameter constraints, (2) *Functional ExpressivE*, where the width parameter of each relation is zero, (3) *EqSlopes ExpressivE*, where all slope vectors are constrained to be equal, (4) *NoCenter ExpressivE*, where the center

[1]https://github.com/AleksVap/ExpressivE

vector of any relation is zero, and (5) *OneBand ExpressivE*, where each relation is embedded by solely one band instead of two. For details and results see Section 4.5.2.

Training Setup. Each model was trained and evaluated on one of four GeForce RTX 2080 GPUs of our internal cluster. Specifically, the training process uses the Adam optimizer (Kingma and Ba, 2015) to optimize the self-adversarial negative sampling loss (Sun et al., 2019). ExpressivE is trained with gradient descent for up to 1000 epochs with early stopping, finishing the training if after 100 epochs the H@10 score did not increase by at least 0.5% for WN18RR and 1% for FB15k-237. We have increased the patience for OneBand ExpressivE to 150 epochs for FB15k-237, as it converges slower than the other ablation versions of ExpressivE. We use the model of the final epoch for testing. Each experiment was repeated three times to account for small performance fluctuations. In particular, the MRR values fluctuate by less than 0.003 between runs for Base and Functional ExpressivE on any dataset. We performed hyperparameter tuning over the learning rate λ, embedding dimensionality d, number of negative samples neg, loss margin γ, adversarial temperature α, and minimal denominator D_{min}. Specifically, two mechanisms were employed to implicitly regularize the hyper-parallelogram: (1) the hyperbolic tangent function $tanh$ was element-wise applied to each entity embedding $\boldsymbol{e_p}$, slope vector $\boldsymbol{s_i^p}$, and center vector $\boldsymbol{c_i^p}$, projecting them into the bounded space $[-1,1]^d$, and (2) the size of each hyper-parallelogram is limited by the novel D_{min} parameter. In the following, we will briefly introduce the D_{min} parameter and its function.

Minimal Denominator. As can be easily shown, Equations 4.109 describe the relation hyper-parallelogram's center, and Equations 4.110–4.111 its corners in $\mathbb{R}^{2d}$.

$$\boldsymbol{center_i^h} = \frac{\boldsymbol{c_i^h} + \boldsymbol{s_i^t c_i^t}}{\boldsymbol{1} - \boldsymbol{s_i^h s_i^t}} \quad \text{and} \quad \boldsymbol{center_i^t} = \frac{\boldsymbol{s_i^h c_i^h} + \boldsymbol{c_i^t}}{\boldsymbol{1} - \boldsymbol{s_i^h s_i^t}} \tag{4.109}$$

$$\boldsymbol{cornA_i^h} = \boldsymbol{center_i^h} \pm \frac{\boldsymbol{w_i^h} + \boldsymbol{s_i^t w_i^t}}{\boldsymbol{1} - \boldsymbol{s_i^h s_i^t}} \quad \text{and} \quad \boldsymbol{cornA_i^t} = \boldsymbol{center_i^t} \pm \frac{\boldsymbol{s_i^h w_i^h} + \boldsymbol{w_i^t}}{\boldsymbol{1} - \boldsymbol{s_i^h s_i^t}} \tag{4.110}$$

$$\boldsymbol{cornB_i^h} = \boldsymbol{center_i^h} \pm \frac{\boldsymbol{w_i^h} - \boldsymbol{s_i^t w_i^t}}{\boldsymbol{1} - \boldsymbol{s_i^h s_i^t}} \quad \text{and} \quad \boldsymbol{cornB_i^t} = \boldsymbol{center_i^t} \pm \frac{\boldsymbol{s_i^h w_i^h} - \boldsymbol{w_i^t}}{\boldsymbol{1} - \boldsymbol{s_i^h s_i^t}} \tag{4.111}$$

Note that the denominator of each term is equal to $(\boldsymbol{1} - \boldsymbol{s_i^h s_i^t})$. Since a small denominator in Equations 4.110 and 4.111 produces large corners and, therefore, a large hyper-parallelogram, we have introduced the hyperparameter D_{min}, allowing ExpressivE to tune the maximal size of its hyper-parallelograms. In particular, D_{min} constrains the relation embeddings such that $(\boldsymbol{1} - \boldsymbol{s_i^h s_i^t}) \succeq D_{min}$, thereby constraining the maximal size of a hyper-parallelogram as required.

Hyperparameter Optimization. Following Abboud et al. (2020), we have varied the learning rate by $\lambda \in \{a * 10^{-b} | a \in \{1,2,5\} \wedge b \in \{-2,-3,-4,-5,-6\}\}$, the margin m by integer values between 3 and 24 inclusive, the adversarial temperature by $\alpha \in \{1,2,3,4\}$, and the number of negative samples by $neg \in \{50,100,150\}$. Furthermore, we have varied the novel minimal denominator parameter by $D_{min} \in \{0, 0.5, 1\}$. We have tuned the hyperparameters of ExpressivE manually within the specified ranges. Finally, to allow a direct performance comparison of ExpressivE to its closest spatial relative BoxE and its closest functional relative RotatE, we chose for each benchmark the embedding dimensionality and negative sampling strategy of the best-performing RotatE and BoxE model (Abboud et al., 2020; Sun et al., 2019). Concretely, we chose self-adversarial negative sampling (Sun et al., 2019) and the embedding dimensionalities listed in Table 4.4. The best-performing hyperparameters for ExpressivE on each benchmark dataset are also listed in Table 4.4. We have used the hyperparameters of Table 4.4 for any considered version of ExpressivE — namely Base, Functional, EqSlopes, NoCenter, and OneBand ExpressivE —, which are described in the ablation study of Section 4.5.2.

Dataset	Embedding Dimensionality	Margin	Learning Rate	Adversarial Temperature	Negative Samples	Batch Size	Minimal Denominator
WN18RR	500	3	$1 * 10^{-3}$	2	100	512	0
FB15k-237	1000	4	$1 * 10^{-4}$	4	150	1024	0.5

Table 4.4: Best hyperparameters for ExpressivE on WN18RR and FB15k-237.

Evaluation. In all of our experiments we: (*i*) employ the standard benchmarks, (*ii*) follow the standard evaluation protocol, and (*iii*) present the standard metrics (filtered MRR and H@k) for KGC. Each of these parts is described in detail in Section 2.1.1.

CO_2 Emission. The computation of the reported experiments took below 200 GPU hours. On an RTX 2080 (TDP of 215W) with a carbon efficiency of 0,432 kg/kWh (based on the OECD's 2014 yearly carbon efficiency average), 200 GPU hours correspond to a rough CO_2 emission of 18.58 kg CO_2-eq. The estimations were conducted using the Machine Learning Emissions Calculator (Lacoste et al., 2019).

4.5 Experimental Evaluation and Space Complexity

In this section, we evaluate ExpressivE on the standard KGC benchmarks WN18RR (Dettmers et al., 2018) and FB15k-237 (Toutanova and Chen, 2015) and report state-of-the-art results, providing strong empirical evidence for the theoretical strengths of ExpressivE (Section 4.5.1). Furthermore, we perform an ablation study on ExpressivE's parameters to quantify the importance of each parameter (Section 4.5.2). Using a relation-wise performance comparison on WN18RR, we analyze the performance of ExpressivE compared to the state of the art (Section 4.5.3). Next, we analyze ExpressivE's performance on relations of various cardinalities to provide empirical evidence for ExpressivE's capability to represent 1–1, 1–N, N–1, and N–N relations (Section 4.5.4). Moreover, we empirically validate that ExpressivE captures general composition rules exactly and exclusively and analyze the link between these results and ExpressivE's performance gain on WN18RR (Section 4.5.5). Finally, we investigate whether ExpressivE can reason over more than one step of composition rules (Section 4.5.6).

4.5.1 Knowledge Graph Completion

Baselines. As in Abboud et al. (2020), we compare ExpressivE to the functional models TransE (Bordes et al., 2013) and RotatE (Sun et al., 2019), spatial model BoxE (Abboud et al., 2020), and bilinear models DistMult (Yang et al., 2015a), ComplEx (Trouillon et al., 2016), and TuckER (Balazevic et al., 2019b). We maintain the fairness of our result comparison by considering gKGEs with a dimensionality $d \leq 1000$ (Balazevic et al., 2019b; Abboud et al., 2020). To allow a direct comparison of ExpressivE's performance and parameter efficiency to its closest functional relative RotatE and spatial relative BoxE, we employ the same dimensionality for the benchmarks as RotatE and BoxE.

Benchmark	Dimensionality	ExpressivE	BoxE	RotatE
WN18RR	500	**467MB**	930MB	930MB
FB15k-237	1000	**366MB**	687MB	687MB

Table 4.5: Model sizes of ExpressivE, BoxE, and RotatE models of equal dimensionality.

Space Complexity. For a d-dimensional embedding, RotatE and BoxE require $(2|\boldsymbol{E}| + 2|\boldsymbol{R}|)d$, whereas ExpressivE solely requires $(|\boldsymbol{E}| + 6|\boldsymbol{R}|)d$ parameters, where $|\boldsymbol{E}|$ is the number of entities and $|\boldsymbol{R}|$ the number of relations. Since $|\boldsymbol{R}| << |\boldsymbol{E}|$ in most graphs, (e.g., FB15k-237: $|\boldsymbol{R}|/|\boldsymbol{E}| = 0.016$) ExpressivE almost *halves* the number of parameters for a d-dimensional embedding compared to BoxE and RotatE. Table 4.5 lists the model sizes of trained ExpressivE, BoxE, and RotatE models of the same dimensionality, empirically confirming that ExpressivE almost *halves* BoxE's and RotatE's sizes.

Family	Model	WN18RR				FB15k-237			
		H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR
Func. & Spatial	Base ExpressivE	**.464**	**.522**	.597	**.508**	.243	.366	.512	.333
	Func. ExpressivE	.407	.519	**.619**	.482	**.256**	**.387**	.535	**.350**
	BoxE	.400	.472	.541	.451	.238	.374	**.538**	.337
	RotatE	.428	.492	.571	.476	.241	.375	.533	.338
	TransE	.013	.401	.529	.223	.233	.372	.531	.332
Bilinear	DistMult	-	-	.531	.452	-	-	.531	.343
	ComplEx	-	-	**.547**	**.475**	-	-	.536	.348
	TuckER	**.443**	**.482**	.526	.470	**.266**	**.394**	**.544**	**.358**

Table 4.6: KGC performance of ExpressivE and state-of-the-art gKGEs on FB15k-237 and WN18RR. The table shows the best-published results of the competing models per family, specifically: TransE and RotatE (Sun et al., 2019), BoxE (Abboud et al., 2020), DistMult and ComplEx (Ruffinelli et al., 2020; Yang et al., 2015b), and TuckER (Balazevic et al., 2019b).

Benchmark Results. We use two versions of ExpressivE in the benchmarks, one where the width parameter $\boldsymbol{w_i}$ is learned and one where $\boldsymbol{w_i} = 0$, called Base ExpressivE and

Functional ExpressivE. Tables 4.5 and 4.6 reveal that Functional ExpressivE, with only *half* the number of parameters of BoxE and RotatE, performs best among spatial and functional models on FB15k-237 and is competitive with TuckER, especially in MRR. Even more, Base ExpressivE *outperforms all* competing models significantly on WN18RR. The significant performance increase of Base ExpressivE on WN18RR is likely due to WN18RR containing both hierarchy and composition rules in contrast to FB15k-237 (similar to the discussion of Abboud et al. (2020)). We will empirically investigate the reasons for ExpressivE's performances on FB15k-237 and WN18RR in Section 4.5.2 and Section 4.5.3.

Discussion. Tables 4.5 and 4.6 reveal that ExpressivE is highly parameter efficient compared to related spatial and functional models while reaching competitive performance on FB15k-237 and even new state-of-the-art performance on WN18RR, supporting the extensive theoretical results of our paper.

4.5.2 Ablation Study

This section analyzes how constraints on ExpressivE's parameters impact its benchmark performances. Specifically, we analyze the following constrained ExpressivE versions: (1) *Base ExpressivE*, which represents ExpressivE without any parameter constraints, (2) *Functional ExpressivE*, where the width parameter $\boldsymbol{w_i}$ of each relation r_i is zero, (3) *EqSlopes ExpressivE*, where all slope vectors are constrained to be equal — i.e., $\boldsymbol{s_i} = \boldsymbol{s_k}$ for any relations r_i and r_k, (4) *NoCenter ExpressivE*, where the center vector $\boldsymbol{c_i}$ of any relation r_i is zero, and (5) *OneBand ExpressivE*, where each relation is embedded by solely one band instead of two — i.e., OneBand ExpressivE captures a triple $r_i(e_h, e_t)$ to be true if its relation and entity embeddings only satisfy Inequality 4.1.

Model	WN18RR				FB15k-237			
	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR
Base ExpressivE	**.464**	**.522**	.597	**.508**	.243	.366	.512	.333
Func. ExpressivE	.407	.519	**.619**	.482	**.256**	**.387**	**.535**	**.350**
EqSlopes ExpressivE	.254	.415	.528	.353	.237	.361	.510	.328
NoCenter ExpressivE	.457	.514	.591	.501	.224	.349	.494	.314
OneBand ExpressivE	.435	.480	.538	.470	.230	.352	.491	.318

Table 4.7: Ablation study on ExpressivE's parameters.

Ablation Results. Table 4.7 provides the results of the ablation study on WN18RR and FB15k-237. It reveals that each component of ExpressivE is vital as setting all slopes $\boldsymbol{s_i}$ to be equal (EqSlopes ExpressivE) or removing the center $\boldsymbol{c_i}$ (NoCenter ExpressivE), width $\boldsymbol{w_i}$ (Functional ExpressivE), or a band (OneBand ExpressivE) results in performance losses on at least one benchmark. Interestingly, Functional outperforms Base ExpressivE on FB15k-237. The reasons for Functional and Base ExpressivE's strong performance on FB15k-237 and, respectively, WN18RR are analyzed in the following:

- **Functional ExpressivE.** Since Functional ExpressivE sets $\boldsymbol{w_i} = 0$, the relation embeddings reduce from a hyper-parallelogram to a function. Intuitively, this means that Functional ExpressivE loses the spatial capabilities of Base ExpressivE, such as the ability to capture hierarchy, while it maintains functional capabilities, such as the ability to capture compositional definition. The effect of the reduced expressive power of Functional ExpressivE can be seen in the performance drop on WN18RR over Base ExpressivE in Table 4.6. However, since Functional ExpressivE uses fewer parameters than Base ExpressivE, it has a lower degree of freedom, making it less likely to stop in a local minimum than Base ExpressivE as empirically supported by Functional ExpressivE's performance gain over Base ExpressivE on FB15k-237 in Table 4.6. Furthermore, the performance gain of Functional ExpressivE over Base ExpressivE on FB15k-237 hints at the dataset not containing many hierarchy rules. Thus, FB15k-237 cannot exploit the added capabilities of Base ExpressivE, namely the ability to capture general composition and hierarchy.

- **Base ExpressivE.** In contrast, Base ExpressivE provides the full expressive power - the high degree of freedom heightening the chance of ending in a local minimum. Table 4.6 reveals a significant performance increase of Base ExpressivE over Functional ExpressivE on WN18RR, giving evidence that the added expressive power can be helpful on datasets containing many composition and hierarchy rules such as WN18RR ((Abboud et al., 2020), cf. Section 4.5.5). The downside of Base ExpressivE's added expressivity is, however, that its higher degrees of freedom may make it likelier to stop in a local optimum, manifesting in its performance drop over Functional ExpressivE on FB15k-237.

4.5.3 WN18RR Performance Analysis

This section analyzes the performance of ExpressivE and its closest spatial relative BoxE (Abboud et al., 2020) and functional relative RotatE (Sun et al., 2019) on WN18RR. Table 4.8 lists the MRR of ExpressivE, RotatE, and BoxE for each of the 11 relations of WN18RR. Bold values represent the best, and underlined values represent the second-best results across the compared models.

Results. ExpressivE performs very well on many relations, where either only BoxE or only RotatE produces good rankings, empirically confirming that ExpressivE combines the inference capabilities of BoxE (hierarchy) and RotatE (compositional definition). Additionally, ExpressivE does not only reach similar performances as RotatE and BoxE if only one of them produces good rankings but even surpasses both of them significantly on relations such as *verb_group*, *also_see*, and *hypernym*. This gives strong experimental evidence that ExpressivE combines the inference capabilities of functional and spatial models, even extending them by novel capabilities (such as general composition), empirically supporting our extensive theoretical results of Section 4.2.

Relation Name	ExpressivE	RotatE	BoxE
member_meronym	**0.233**	0.199	<u>0.226</u>
hypernym	**0.189**	<u>0.162</u>	0.159
has_part	**0.198**	<u>0.187</u>	0.168
instance_hypernym	<u>0.352</u>	0.326	**0.425**
synset_domain_topic_of	<u>0.363</u>	**0.384**	0.323
member_of_domain_usage	0.288	<u>0.333</u>	**0.360**
member_of_domain_region	0.123	<u>0.188</u>	**0.189**
also_see	**0.649**	<u>0.631</u>	0.517
derivationally_related_from	**0.956**	<u>0.943</u>	0.902
similar_to	**1.000**	**1.000**	**1.000**
verb_group	**0.972**	0.843	<u>0.876</u>

Table 4.8: Relation-wise MRR comparison of ExpressivE, RotatE, and BoxE on WN18RR.

4.5.4 Cardinality Experiments

This section studies the benchmark performances of ExpressivE and its closest relatives on WN18RR stratified by the cardinality of each relation, providing empirical evidence that ExpressivE performs well on 1–1, 1–N, N–1, and N–N relations.

Experiment Setup. Following the procedure of Bordes et al. (2013), we have categorized the relations of WN18RR into four cardinality classes, specifically 1–1, 1–N, N–1, and N–N. As in Bordes et al. (2013), we have classified a relation $r \in \boldsymbol{R}$ by computing:

- μ_{rt} the averaged number of head entities $h \in \boldsymbol{E}$ per tail entity $t \in \boldsymbol{E}$, appearing in a triple $r(h, t)$ of WN18RR.
- μ_{rh} the averaged number of tail entities $t \in \boldsymbol{E}$ per head entity $h \in \boldsymbol{E}$, appearing in a triple $r(h, t)$ of WN18RR.

Following the soft classification of Bordes et al. (2013), a relation is:

- **1–1** if $\mu_{rt} \leq 1.5$ and $\mu_{rh} \leq 1.5$
- **1–N** if $\mu_{rt} \leq 1.5$ and $\mu_{rh} \geq 1.5$
- **N–1** if $\mu_{rt} \geq 1.5$ and $\mu_{rh} \leq 1.5$
- **N–N** if $\mu_{rt} \geq 1.5$ and $\mu_{rh} \geq 1.5$

Results. Table 4.9 summarizes the performance results of ExpressivE and its closest spatial relative BoxE and functional relative RotatE on WN18RR, stratified by the four cardinality classes defined previously. It reveals that ExpressivE almost exclusively reaches a competitive or state-of-the-art performance on 1–N, N–1, and N–N relations. In particular, ExpressivE outperforms both RotatE and BoxE consistently on N–N relations, which are often considered the most complex relations to capture in KGC with regard to cardinalities. Thus, Table 4.9 provides empirical results supporting our theoretical claim that ExpressivE can capture 1–1, 1–N, N–1, and N–N relations well.

Task	**Predicting Head**				**Predicting Tail**			
Cardinality	1–1	1–N	N–1	N–N	1–1	1–N	N–1	N–N
ExpressivE	**0.976**	<u>0.290</u>	<u>0.105</u>	**0.941**	**0.976**	<u>0.141</u>	**0.327**	**0.938**
RotatE	0.833	**0.294**	0.103	<u>0.930</u>	0.875	0.107	<u>0.288</u>	<u>0.925</u>
BoxE	<u>0.877</u>	0.272	**0.146**	0.883	<u>0.893</u>	**0.147**	0.246	0.884

Table 4.9: MRR of ExpressivE, RotatE, and BoxE on WN18RR by cardinality class (1–1, 1–N, N–1, N–N). The best results are bold, and the second-best are underlined.

4.5.5 General Composition and Link to Performance Gain

This section provides empirical evidence for the theoretical result of Sections 4.2.6 and 4.2.7 that ExpressivE can capture general composition exactly and exclusively. Even more, the experiments in this section provide evidence for a direct link between the support of general composition and ExpressivE's performance gain on WN18RR. In the following, we first discuss the details of our experiments' preparation and setup, followed by the considered hypotheses and final results.

Rule Identification. Our first goal, to provide empirical evidence for the discussed points, was to identify rules occurring in WN18RR. To reach this goal, we have analyzed rules mined with AMIE+ (Galárraga et al., 2015) from WN18RR by Akrami et al. (2020) that were provided in a GitHub repository[2]. To identify the most relevant rules, we have — similar to the discussion of (Galárraga et al., 2013, 2015) — sorted the rules $\rho = \phi_{B1} \wedge \cdots \wedge \phi_{Bm} \Rightarrow r(X,Y)$ by their head coverage $h(\rho)$, which is formally defined as (Galárraga et al., 2013):

$$h(\rho) = \frac{|\{(x,y) \in \boldsymbol{E}^2 \mid r(x,y) \in \boldsymbol{G} \wedge \exists z_1 \dots z_k(\phi_{B1}(z_1,z_2) \in \boldsymbol{G} \wedge \cdots \wedge \phi_{Bm}(z_{k-1},z_k) \in \boldsymbol{G})\}|}{|\{(x,y) \in \boldsymbol{E}^2 \mid r(x,y) \in \boldsymbol{G}\}|}$$

On an intuitive level, the head coverage $h(\rho)$ represents the ratio of true triples implied by the rule ρ on a given Knowledge Graph $(\boldsymbol{G}, \boldsymbol{E}, \boldsymbol{R})$.

Rule Selection. To analyze the most relevant rules in the following experiments, we have selected any rules whose head coverage is greater than 15% (as inspection of the head coverage of AMIE shows a very low number of inferred triples contained in the test set below that). From these rules, we have left out any rule with the head relation *_similar_to*, as ExpressivE, BoxE, and RotatE already have an MRR of 1 on this relation,

[2]https://github.com/idirlab/kgcompletion (last visited 07/25/2024)

thus further stratifying *_similar_to*'s test triples will not reveal novel information. This procedure leads to the following set of rules, where relations of the form r^{-1} depict the inverse relation of $r \in \boldsymbol{R}$:

$$\begin{aligned}
S_1 &:= \_verb\_group(Y,X) \Rightarrow \_verb\_group(X,Y)\\
C_2 &:= \_derivationally\_related\_form(X,Y) \wedge\\
&\quad\ \_derivationally\_related\_form(Y,Z) \Rightarrow \_verb\_group(X,Z)\\
C_3 &:= \_derivationally\_related\_form(X,Y) \wedge\\
&\quad\ \_derivationally\_related\_form^{-1}(Y,Z) \Rightarrow \_verb\_group(X,Z)\\
C_4 &:= \_derivationally\_related\_form^{-1}(X,Y) \wedge\\
&\quad\ \_derivationally\_related\_form(Y,Z) \Rightarrow \_verb\_group(X,Z)\\
C_5 &:= \_also\_see(X,Y) \wedge \_also\_see(Y,Z) \Rightarrow \_also\_see(X,Z)\\
C_6 &:= \_also\_see(X,Y) \wedge \_also\_see^{-1}(Y,Z) \Rightarrow \_also\_see(X,Z)\\
S_7 &:= \_also\_see(Y,X) \Rightarrow \_also\_see(X,Y)\\
C_8 &:= \_hypernym(X,Y) \wedge\\
&\quad\ \_synset\_domain\_topic\_of(Y,Z) \Rightarrow \_synset\_domain\_topic\_of(X,Z)
\end{aligned}$$

Experimental Setup. For each of these rules ρ, we have computed all triples that (i) can be derived by ρ from the data known to our model and (ii) are known to be true in the KG, yet unseen to our models. Thus, for each rule ρ, we have computed the set s_ρ, containing all triples that (i) can be derived with ρ from the training set and (ii) are contained in the test set of WN18RR. We have used each of the computed sets of triples s_ρ to evaluate the performance of ExpressivE, BoxE, and RotatE on the corresponding rule ρ.

Hypotheses. Note that (as discussed in Section 4.3.1) compositional definition $r_1(X,Y) \wedge r_2(Y,Z) \Leftrightarrow r_3(X,Z)$ defines the triples of the composite relation r_3 completely, whereas general composition $r_1(X,Y) \wedge r_2(Y,Z) \Rightarrow r_3(X,Z)$ allows r_3 to contain more triples than those that the compositional definition rule can directly infer. Thus, if ExpressivE captures general composition and if RotatE captures compositional definition, we expect the following behavior:

- **H1.** RotatE will perform well solely on relations occurring as the head of maximally one composition rule, as RotatE solely supports compositional definition.
- **H2.** ExpressivE will perform well even when a relation is defined by multiple composition rules and multiple other rules since ExpressivE supports general composition.

Results. Table 4.10 lists for each rule S_1 to C_8 the performances of BoxE, RotatE, and ExpressivE on s_ρ, where $\rho \in \{S_1, \dots, C_8\}$ and where S_i represents a symmetry rule and C_i represents a composition rule. Table 4.10 provides evidence for both hypotheses:

- **Evidence for H1.** In the case of the relation *_synset_domain_topic_of* (*_syn_dto*), there is only one rule that has *_synset_domain_topic_of* as its head relation, specifically the composition rule C_8. RotatE achieves comparable performance to ExpressivE on s_{C_8} as RotatE is capable of defining *_synset_domain_topic_of* using compositional definition, providing evidence for H1.

- **Evidence for H2.** Yet, when a relation is defined via multiple rules, RotatE's performance decreases drastically on most composition rules compared to ExpressivE's performance, as can be seen for the rules C_2, C_3, C_4, and C_5, giving evidence for hypothesis H2.

Head Rel.	**_verb_group**				**_also_see**			**_syn_dto**
Model	S_1	C_2	C_3	C_4	C_5	C_6	S_7	C_8
Base Exp.	**1.000**	**1.000**	**1.000**	**1.000**	**0.818**	**0.907**	**0.985**	**0.621**
RotatE	0.865	0.760	0.760	0.760	0.771	0.893	0.975	0.599
BoxE	0.906	0.801	0.806	0.806	0.632	0.645	0.727	0.547

Table 4.10: MRR of ExpressivE, RotatE, and BoxE on WN18RR stratified by rules S_1-C_8. S_i represents a [S]ymmetry rule, C_i a [C]omposition rule ($i \in \{1, \dots, 8\}$).

Conclusion. Thus, these experiments empirically support that (i) ExpressivE can capture general composition, as ExpressivE and RotatE perform as expected by H1 and H2, i.e., under the assumption that ExpressivE captures general composition, while RotatE solely captures compositional definition. Furthermore, the experiments support that (ii) ExpressivE's ability to capture general composition contributes to the performance gain on WN18RR, as ExpressivE consistently outperforms RotatE and BoxE on triples inferred from composition rules.

4.5.6 Multiple Steps of Composition

This section presents empirical results for ExpressivE's ability to reason over more than one step of composition rules, formally proven in Section 4.2.8. To evaluate how well ExpressivE supports more than one step of a composition rule, our first goal was to identify multi-step rules (i.e., rules that can be "chained" in multiple steps) occurring in WN18RR. We recall parts of Section 4.5.5 for the self-containedness of this section, readers who have read that section can skip ahead to the "experimental setup" paragraph. To reach the goal of identifying multi-step rules occurring in WN18RR, we have analyzed rules mined with AMIE+ (Galárraga et al., 2015) from WN18RR by Akrami et al. (2020) that were provided in a GitHub repository[3]. To identify the most relevant rules, we have — similar to (Galárraga et al., 2013, 2015) — sorted the rules $\rho = \phi_{B1} \wedge \cdots \wedge \phi_{Bm} \Rightarrow r(X, Y)$ by their head coverage $h(\rho)$, which is formally defined as (Galárraga et al., 2013):

[3]https://github.com/idirlab/kgcompletion (last visited 07/25/2024)

$$h(\rho) = \frac{|\{(x,y) \in \boldsymbol{E}^2 \mid r(x,y) \in \boldsymbol{G} \wedge \exists z_1 \dots z_k(\phi_{B1}(z_1,z_2) \in \boldsymbol{G} \wedge \dots \wedge \phi_{Bm}(z_{k-1},z_k) \in \boldsymbol{G})\}|}{|\{(x,y) \in \boldsymbol{E}^2 \mid r(x,y) \in \boldsymbol{G}\}|}$$

At an intuitive level, the head coverage $h(\rho)$ represents the ratio of true triples implied by the rule ρ on a given Knowledge Graph $(\boldsymbol{G}, \boldsymbol{E}, \boldsymbol{R})$.

Next, we present the four multi-step rules with head coverage of at least 15%, as discussed in Section 4.5.5:

$$\begin{aligned} R_1 &:= \_hypernym(X,Y) \wedge \\ &\qquad \_synset\_domain\_topic\_of(Y,Z) \Rightarrow \_synset\_domain\_topic\_of(X,Z) \\ R_2 &:= \_also\_see(X,Y) \wedge \_also\_see(Y,Z) \Rightarrow \_also\_see(X,Z) \\ R_3 &:= \_also\_see(X,Y) \wedge \_also\_see^{-1}(Y,Z) \Rightarrow \_also\_see(X,Z) \\ R_4 &:= \_also\_see^{-1}(X,Y) \wedge \_also\_see^{-1}(Y,Z) \Rightarrow \_also\_see(X,Z) \end{aligned}$$

The relation $\_also\_see^{-1}$ of R_3 and R_4 represents the inverse relation of $\_also\_see$.

Experimental Setup. For each of the selected multi-step rules $\rho \in \{R_1, R_2, R_3, R_4\}$, we have generated three datasets, the *1-Step*, *2-Steps*, and *3-Steps* sets. Specifically, we have generated for each ρ a *j-Step(s)* set by computing all triples that (i) can be derived by ρ in j steps from the data known to our model and (ii) are known to be true in the KG, yet unseen to our model. Thus, we have computed for each ρ a *j-Step(s)* set, containing all triples that (i) can be derived with ρ by j applications on the training set and (ii) are contained in the test set of WN18RR. The performance of ExpressivE on the computed datasets is summarised in Table 4.11.

	1-Step	*2-Steps*	*3-Steps*	*4-Steps+*
R_1	**0.627**	0.621	-	-
R_2	0.720	0.804	**0.818**	-
R_3	0.768	**0.907**	-	-
R_4	0.716	**0.922**	-	-

Table 4.11: ExpressivE's MRR on WN18RR in dependence on the number of reasoning steps. Hyphens represent that no new triples can be inferred with additional steps.

Results. We report the performance of at most two steps of $R_1/R_3/R_4$ as after applying $R_1/R_3/R_4$ twice on the training set; no new triples are derived. Similarly, no new triples are derived after at most three steps of R_2 on the training set. We can see that the performance of ExpressivE increases by a large margin when more than one step of reasoning is considered, depicted by the performance gain of the 2-Steps and 3-Steps set over the 1-Step set. Interestingly, a small exception for this is R_1, where we see a slightly worse behavior – inspection of the results shows that this is due to a single triple. In total, Table 4.11 provides empirical evidence that ExpressivE can capture chained composition rules and thus perform more than one step of reasoning.

4.6 Summary

In this chapter, we have reached our first research goal (G1) by introducing ExpressivE, a gKGE that — as required by G1 — (*i*) captures all core inference rules, including hierarchy and general composition, which are fundamental rules for representing data properties in the DB and SW fields. Thereby, our model overcomes the reasoning divide, strengthening the connection between ML-, DB- and SW-based KG research. Additionally, ExpressivE (*ii*) represents inference rules through spatial relations of hyper-parallelograms, offering an intuitive and consistent geometric interpretation of ExpressivE embeddings and their captured rules, (*iii*) is fully expressive, and (*iv*) reaches competitive performance on FB15k-237, even outperforming any competing gKGE significantly on WN18RR.

CHAPTER 5

Scalability Divide

As discussed in Section 1.2, recall that most KGEs suffer from at least one of the following efficiency problems: They rely either on (i) a high embedding dimensionality to reach state-of-the-art KGC performance, limiting their scalability or (ii) a more complex embedding space, typically requiring computationally more costly operations or a higher number of parameters, increasing their space and time requirements.

Facing these challenges, this chapter designs a *Euclidean* gKGE that performs well on KGC with *low-dimensional* embeddings, reducing its storage space, inference, and training times and, thereby reaching G2 (defined in Section 1.2). To reach this goal, we perform simple yet impactful enhancements on the parameters and distance function of ExpressivE — a Euclidean gKGE that has shown state-of-the-art performance on KGC with high-dimensional embeddings (see Chapter 4 for details) — raising its efficiency. Based on our ExpressivE model (introduced in Chapter 4):

- We propose the lightweight SpeedE model that halves ExpressivE's inference time and enhances ExpressivE's distance function, significantly improving its KGC performance.
- We evaluate SpeedE on the three standard KGC benchmarks, WN18RR, FB15k-237, and YAGO3-10, finding that it is competitive with state-of-the-art gKGEs on FB15k-237 and even outperforms them significantly on WN18RR and the large YAGO3-10 benchmark.
- We find that SpeedE preserves ExpressivE's KGC performance on WN18RR with much fewer parameters, in particular, requiring solely a fourth of the number of parameters of ExpressivE and solely a fifth of its training time to reach the same KGC performance (as we shall see in Section 5.3.3).

In total, we reach G2 by proposing the highly scalable SpeedE model, which reaches strong KGC performance using low-dimensional embeddings while maintaining the low space and time requirements of Euclidean gKGEs.

Organization. This chapter contains material from the following constituent papers of this dissertation (Pavlović and Sallinger, 2023a,b, 2024a,b,c). Based on these publications, Section 5.1 disassembles ExpressivE's components to find a simpler model that still supports the core inference rules (which are defined in Chapter 2) and continues by building on these results to introduce the lightweight SpeedE model. Before proceeding to empirical results, Section 5.2 discusses the experimental setup, listing details on reproducing our results, SpeedE's implementation, training setup, evaluation protocol, and estimated CO_2 emissions. Next, Section 5.3 empirically evaluates SpeedE's KGC performance, studies its space and time efficiency, and analyzes the relevance of SpeedE's novel distance slope parameters by performing an ablation study. Section 5.4 summarizes all of our results.

5.1 The Methodology

Our goal is to design a KGE that addresses the efficiency problems raised by the use of (1) complex embedding spaces and (2) high-dimensional embeddings while (3) allowing for a geometric interpretation of its embeddings (Abboud et al., 2020; Pavlović and Sallinger, 2023b). We reach this goal by designing a KGC model that (1) is based in the Euclidean space, (2) reaches high KGC performance under low-dimensional conditions while at the same time supports all *core inference rules*, and (3) is a gKGE.

Toward our goal, Section 5.1.1 analyzes the ExpressivE model, finding that it uses redundant parameters that negatively affect its inference time. By redundant parameters, we mean parameters that can be removed while preserving the support of the core inference rules. Facing this problem, we propose the lightweight Min_SpeedE model that removes these redundancies, halving ExpressivE's inference time (Section 5.1.1).

However, Min_SpeedE loses the ability to adjust its distance function, which is essential for representing hierarchical relations (as empirically verified in Section 5.3). Thus, Section 5.1.2 introduces SpeedE, a model that enhances Min_SpeedE by adding carefully designed parameters for flexibly adjusting the distance function while preserving Min_SpeedE's low inference times. Since our formal definitions (Section 5.1.3) and proofs (Section 5.1.4) are quite technical and long, we provide them at the end of this section.

5.1.1 Min_SpeedE

To design Min_SpeedE, let us first analyze ExpressivE's parameters, particularly its width vector. Adjusting ExpressivE's width vector $\boldsymbol{w_i}$ has two competing effects: (1) it alters the distance function's slopes (by $\boldsymbol{d_i}$ in Equation 4.3), and (2) it changes which entity pairs are inside the relation hyper-parallelogram (by $\boldsymbol{w_i}$ in the inequality condition $\boldsymbol{\tau}_{r_i(h,t)} \preceq \boldsymbol{w_i}$ of Equation 4.3). To increase ExpressivE's time efficiency substantially, we introduce Min_SpeedE, a constrained version of ExpressivE that replaces the relation-wise width vectors $\boldsymbol{w_i} \in (\mathbb{R}_{\geq 0})^{2d}$ by a constant value $w \in \mathbb{R}_{>0}$ - that is shared across all relations $r_i \in \boldsymbol{R}$. The following paragraphs theoretically analyze Min_SpeedE's inference capabilities and time efficiency.

Inference Capabilities. We find that Min_SpeedE surprisingly still captures the core inference rules and prove this in Theorem 5.1.1. We give the full proof in Section 5.1.4 and discuss one of the most interesting parts here, namely, hierarchy rules.

Theorem 5.1.1. *Min_SpeedE captures the core inference rules, i.e., symmetry, anti-symmetry, inversion, general composition, hierarchy, intersection, and mutual exclusion.*

Hierarchy rules. Recall from Section 4.2.2 that an ExpressivE model captures a hierarchy rule $r_1(X, Y) \Rightarrow r_2(X, Y)$ iff r_1's hyper-parallelogram is a proper subset of r_2's. Thus, one would expect that ExpressivE's ability to capture hierarchy rules is lost in Min_SpeedE, as the width parameter $w \in \mathbb{R}_{>0}$ (responsible for adjusting a hyper-parallelogram's size) is shared across all hyper-parallelograms. However, the actual size of a hyper-parallelogram does not solely depend on its width but also on its slope parameter $\boldsymbol{s_i} \in \mathbb{R}^{2d}$, allowing one hyper-parallelogram $\boldsymbol{H_1}$ to properly subsume another $\boldsymbol{H_2}$ even when they share the same width parameter w. We have visualized two hyper-parallelograms $\boldsymbol{H_2} \subset \boldsymbol{H_1}$ with the same width parameter w in Figure 5.1.

Intuition. Min_SpeedE can capture $\boldsymbol{H_2} \subset \boldsymbol{H_1}$ as w (depicted with orange dotted lines) represents the intersection of the bands (depicted with blue and green dotted lines), expanded from the hyper-parallelogram, and the axis of the band's corresponding dimension. Thus, a hyper-parallelogram's actual size can be adapted by solely changing its slopes, removing the need for a learnable width parameter per dimension and relation.

Inference Time. The most costly operations during inference are operations on vectors. Thus, we can estimate ExpressivE's and Min_SpeedE's inference time by counting the number of vector operations necessary for computing a triple's score: By reducing the width vector to a scalar, many operations reduce from a vector to a scalar operation. In particular, the calculation of $\boldsymbol{d_i}$ and $\boldsymbol{k_i}$ uses solely scalars in Min_SpeedE instead of vectors (c.f., Section 4.1). Thus, ExpressivE needs 15, whereas Min_SpeedE needs solely 8 vector operations to compute a triple's score. This corresponds to Min_SpeedE using approximately half the number of vector operations of ExpressivE for computing a triple's score, thus roughly halving ExpressivE's inference time, which aligns with Section 5.3.3's empirical results.

Key Insights. Fixing the width to a constant value w stops Min_SpeedE from adjusting the distance function's slopes. As we will empirically see in Section 5.3, the effect of this is a severely degraded KGC performance on hierarchical relations. Introducing independent parameters for adjusting the distance function's slopes solves this problem. However, these parameters must be designed carefully to (1) preserve ExpressivE's geometric interpretation and (2) retain the reduced inference time provided by Min_SpeedE. Each of these aspects will be covered in detail in the next section.

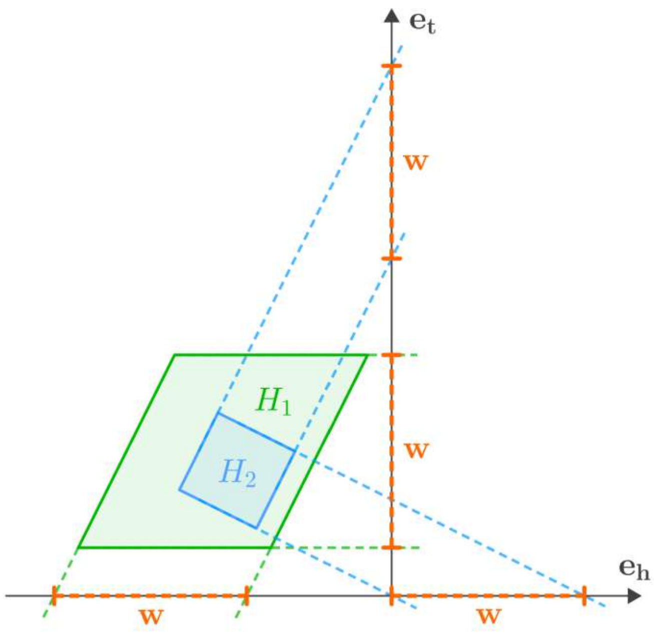

Figure 5.1: Representation of the two-dimensional relation hyper-parallelograms $\boldsymbol{H_1}$ and $\boldsymbol{H_2}$, such that $\boldsymbol{H_1}$ subsumes $\boldsymbol{H_2}$ and such that they share the same width parameter w in each dimension.

5.1.2 SpeedE

SpeedE further enhances Min_SpeedE by adding the following two carefully designed scalar parameters to each relation embedding: (1) the inside distance slope $s_i^\iota \in [0,1]$ and (2) the outside distance slope s_i^o with $s_i^\iota \leq s_i^o$. Let $d_i^\iota := 2s_i^\iota w + 1$, $d_i^o := 2s_i^o w + 1$, and $k_i := d_i^o(d_i^o - 1)/2 - (d_i^\iota - 1)/(2d_i^\iota)$, then SpeedE defines the following distance function:

$$D(h, r_i, t) = \begin{cases} \boldsymbol{\tau_{r_i(h,t)}} \oslash d_i^\iota, & \text{if } \boldsymbol{\tau_{r_i(h,t)}} \preceq w \\ \boldsymbol{\tau_{r_i(h,t)}} \odot d_i^o - k_i, & \text{otherwise} \end{cases} \tag{5.1}$$

Again, the distance function is separated into two piece-wise linear functions: (1) the inside distance $D_\iota(h, r_i, t) = \boldsymbol{\tau_{r_i(h,t)}} \oslash d_i^\iota$ for triples that are captured to be true (i.e., $\boldsymbol{\tau_{r_i(h,t)}} \preceq w$) and (2) the outside distance $D_o(h, r_i, t) = \boldsymbol{\tau_{r_i(h,t)}} \odot d_i^o - k_i$ for triples that are captured to be false (i.e., $\boldsymbol{\tau_{r_i(h,t)}} \npreceq w$). Based on this function, SpeedE defines the score as $s(h, r_i, t) = -||D(h, r_i, t)||_2$.

Geometric Interpretation. The intuition of s_i^ι and s_i^o is that they control the slopes of the respective linear inside and outside distance functions. However, without any constraints on s_i^ι and s_i^o, SpeedE would lose ExpressivE's intuitive geometric interpretation (cf., Section 4.2.2) as s_i^ι and s_i^o could be chosen in such a way that distances of embeddings within the hyper-parallelogram are larger than those outside. By constraining these parameters to $s_i^\iota \in [0,1]$ and $s_i^\iota \leq s_i^o$, we preserve lower distances within hyper-parallelograms than outside and, thereby, the intuitive geometric interpretation.

Inference Time. The additional introduction of two scalar distance slope parameters $s_i^t, s_i^o \in \mathbb{R}$ per relation r_i does not change the number of vector operations necessary for computing a triple's score and, thus, does not significantly affect SpeedE's inference time. Therefore, we expect that SpeedE retains the time efficiency of Min_SpeedE, as empirically validated in Section 5.3.3.

With this, we have finished our introduction of SpeedE. What remains to be shown are the formal definitions of SpeedE (Section 5.1.3) and the proofs of its theoretical capabilities (Section 5.1.4).

5.1.3 Formal Definitions

In this section, we introduce the formal semantics of SpeedE models. Specifically, this section slightly adapts the notions of capturing a rule in an ExpressivE model (formally defined in Section 4.2.3) for the SpeedE model. For the convenience of the reader and to give a quick reference to fall back in this chapter, we provide here the full formal definitions for SpeedE. In what follows, we employ the same formal definition for a Knowledge Graph $(\boldsymbol{G}, \boldsymbol{E}, \boldsymbol{R})$ and the same notations as introduced in Section 4.2.3.

SpeedE Model. We define a SpeedE model as a tuple $\boldsymbol{M^+} = (\boldsymbol{\epsilon}, \boldsymbol{\sigma}, w, \boldsymbol{\rho})$, where $\boldsymbol{\epsilon} \subset 2^{\mathbb{R}^d}$ is the set of entity embeddings, $\boldsymbol{\sigma} \subset 2^{\mathbb{R}^d}$ is the set of center embeddings, $w \in \mathbb{R}_{>0}$ represents the width constant, and $\boldsymbol{\rho} \subset 2^{\mathbb{R}^d}$ is the set of slope vectors.

Linking Embeddings to KGs. A SpeedE model $\boldsymbol{M^+} = (\boldsymbol{\epsilon}, \boldsymbol{\sigma}, w, \boldsymbol{\rho})$ and a KG $(\boldsymbol{G}, \boldsymbol{E}, \boldsymbol{R})$ are linked via the following assignment functions: We employ ExpressivE's entity assignment function $\boldsymbol{f_e} : \boldsymbol{E} \rightarrow \boldsymbol{\epsilon}$ and virtual assignment function $\boldsymbol{f_v} : \boldsymbol{E} \times \boldsymbol{E} \rightarrow \mathbb{R}^{2d}$ directly for the SpeedE model (see Section 4.2.3 for details). For completeness, we recall the definitions of $\boldsymbol{f_e}$ and $\boldsymbol{f_v}$ briefly. The entity assignment function $\boldsymbol{f_e}$ assigns to each entity $e_h \in \boldsymbol{E}$ an entity embedding $\boldsymbol{e_h} \in \boldsymbol{\epsilon}$. Based on $\boldsymbol{f_e}$, the virtual assignment function $\boldsymbol{f_v}$ defines for any pair of entities $(e_h, e_t) \in \boldsymbol{E}$ a virtual entity pair embedding $\boldsymbol{f_v}(e_h, e_t) = (\boldsymbol{f_e}(e_h) || \boldsymbol{f_e}(e_t))$, where $||$ represents concatenation. Next, we define SpeedE's relation assignment function $\boldsymbol{f_h^+}(r_i) : \boldsymbol{R} \rightarrow \mathbb{R}^{2d} \times \mathbb{R} \times \mathbb{R}^{2d}$ as $\boldsymbol{f_h^+}(r_i) = (\boldsymbol{c_i}, w, \boldsymbol{s_i})$, where $\boldsymbol{c_i} = (\boldsymbol{c_i^h} || \boldsymbol{c_i^t})$ with $\boldsymbol{c_i^h}, \boldsymbol{c_i^t} \in \boldsymbol{\sigma}$ and where $\boldsymbol{s_i} = (\boldsymbol{s_i^t} || \boldsymbol{s_i^h})$ with $\boldsymbol{s_i^t}, \boldsymbol{s_i^h} \in \boldsymbol{\rho}$.

Model Configuration. We call a SpeedE model $\boldsymbol{M^+}$ together with a concrete relation assignment function $\boldsymbol{f_h^+}$ a relation configuration $\boldsymbol{m_h^+} = (\boldsymbol{M^+}, \boldsymbol{f_h^+})$. If $\boldsymbol{m_h^+}$ additionally has a virtual assignment function $\boldsymbol{f_v}$, we call it a complete model configuration $\boldsymbol{m^+} = (\boldsymbol{M^+}, \boldsymbol{f_h^+}, \boldsymbol{f_v})$.

Definition of Truth. A triple $r_i(e_h, e_t)$ is captured to be true in some $\boldsymbol{m^+}$, with $r_i \in \boldsymbol{R}$ and $e_h, e_t \in \boldsymbol{E}$ iff Inequality 5.2 holds for the assigned embeddings of h, t, and r. This means more precisely that Inequality 5.2 needs to hold for $\boldsymbol{f_v}(e_h, e_t) = (\boldsymbol{f_e}(e_h) || \boldsymbol{f_e}(e_t)) = (\boldsymbol{e_h}, \boldsymbol{e_t})$ and $\boldsymbol{f_h^+}(r_i) = (\boldsymbol{c_i}, w, \boldsymbol{s_i})$.

$$(\boldsymbol{e_{ht}} - \boldsymbol{c_i} - \boldsymbol{s_i} \odot \boldsymbol{e_{th}})^{|.|} \preceq w, \tag{5.2}$$

Intuition. At an intuitive level, a triple $r_i(e_h, e_t)$ is captured to be true by some complete SpeedE model configuration $\boldsymbol{m}^+$ iff the virtual pair embedding $\boldsymbol{f_v}(e_h, e_t)$ of entities e_h and e_t lies within the hyper-parallelogram of relation r_i defined by $\boldsymbol{f_h^+}(r_i)$.

Simplifying Notations. To simplify the upcoming proofs, we denote with $\boldsymbol{f_v}(e_h, e_t) \in \boldsymbol{f_h^+}(r_i)$ that the virtual pair embedding $\boldsymbol{f_v}(e_h, e_t)$ of an entity pair $(e_h, e_t) \in \boldsymbol{E} \times \boldsymbol{E}$ lies within the hyper-parallelogram $\boldsymbol{f_h^+}(r_i)$ of some relation $r_i \in \boldsymbol{R}$ in the virtual triple space. Accordingly, for sets of virtual pair embeddings $\boldsymbol{P} := \{\boldsymbol{f_v}(e_{h_1}, e_{t_1}), \dots, \boldsymbol{f_v}(e_{h_n}, e_{t_n})\}$, we denote with $\boldsymbol{P} \subseteq \boldsymbol{f_h^+}(r_i)$ that all virtual pair embeddings of $\boldsymbol{P}$ lie within the hyper-parallelogram of the relation r_i. Furthermore, we denote with $\boldsymbol{f_v}(e_h, e_t) \notin \boldsymbol{f_h^+}(r_i)$ that a virtual pair embedding $\boldsymbol{f_v}(e_h, e_t)$ does not lie within the hyper-parallelogram of a relation r_i and with $\boldsymbol{P} \nsubseteq \boldsymbol{f_h^+}(r_i)$ we denote that an entire set of virtual pair embeddings $\boldsymbol{P}$ does not lie within the hyper-parallelogram of a relation r_i.

Capturing Inference rules. Based on the previous definitions, we define capturing rules formally: A relation configuration $\boldsymbol{m_h^+}$ captures a rule ψ *exactly* if for any ground rule $\phi_{B_1} \land \dots \land \phi_{B_m} \Rightarrow \phi_H$ within the deductive closure of ψ and for any instantiation of $\boldsymbol{f_e}$ and $\boldsymbol{f_v}$ the following conditions hold:

- if ϕ_H is a triple and if $\boldsymbol{m_h^+}$ captures the body triples to be true — i.e., $\boldsymbol{f_v}(\mathit{args}(\phi_{B_1})) \in \boldsymbol{f_h^+}(\mathit{rel}(\phi_{B_1})), \dots, \boldsymbol{f_v}(\mathit{args}(\phi_{B_m})) \in \boldsymbol{f_h^+}(\mathit{rel}(\phi_{B_m}))$ — then $\boldsymbol{m_h^+}$ also captures the head triple to be true — i.e., $\boldsymbol{f_v}(\mathit{args}(\phi_H)) \in \boldsymbol{f_h^+}(\mathit{rel}(\phi_H))$.
- if $\phi_H = \bot$, then $\boldsymbol{m_h^+}$ captures at least one of the body triples to be false — i.e., there is some $j \in \{1, \dots, m\}$ such that $\boldsymbol{f_v}(\mathit{args}(\phi_{B_i})) \notin \boldsymbol{f_h^+}(\mathit{rel}(\phi_{B_i}))$.

where *args*() is the function that returns the arguments of a triple, and *rel*() is the function that returns the relation of the triple. Furthermore, a relation configuration $\boldsymbol{m_h^+}$ captures a rule ψ *exactly* and *exclusively* if (1) $\boldsymbol{m_h^+}$ exactly captures ψ and (2) $\boldsymbol{m_h^+}$ does not capture any *positive* rule ϕ (i.e., $\phi \in \{\mathit{symmetry},\ \mathit{inversion},\ \mathit{hierarchy},\ \mathit{intersection},\ \mathit{generalcomposition}\}$) such that $\psi \not\models \phi$ except where the body of ϕ is not satisfied over $\boldsymbol{m_h^+}$.

These formal definitions for capturing rules in SpeedE models are in accordance with the literature (Abboud et al., 2020; Pavlović and Sallinger, 2023b) and follow the same intuition as their counterparts for ExpressivE discussed in Section 4.2.3.

5.1.4 Proof of Theorem 5.1.1

To prove that SpeedE captures the core inference rules exactly and exclusively (Theorem 5.1.1), let us first recall Section 4.2.6's formal definition of these rules (cf., Definition 4.2.8). For the convenience of the reader and to give a quick reference, we restate the definitions of the core inference patterns below (Definition 5.1.2).

Definition 5.1.2. *(Abboud et al., 2020; Pavlović and Sallinger, 2023b) Let the core inference rules be defined as follows:*

- *rules of the form* $r_1(X,Y) \Rightarrow r_1(Y,X)$ *with* $r_1 \in \boldsymbol{R}$ *are called* symmetry rules*.*
- *rules of the form* $r_1(X,Y) \wedge r_1(Y,X) \Rightarrow \bot$ *with* $r_1 \in \boldsymbol{R}$ *are called* anti-symmetry rules*.*
- *rules of the form* $r_1(X,Y) \Leftrightarrow r_2(Y,X)$ *with* $r_1, r_2 \in \boldsymbol{R}$ *and* $r_1 \neq r_2$ *are called* inversion rules*.*
- *rules of the form* $r_1(X,Y) \wedge r_2(Y,Z) \Rightarrow r_3(X,Z)$ *with* $r_1, r_2, r_3 \in \boldsymbol{R}$ *and* $r_1 \neq r_2 \neq r_3$ *are called general* composition rules*.*
- *rules of the form* $r_1(X,Y) \Rightarrow r_2(X,Y)$ *with* $r_1, r_2 \in \boldsymbol{R}$ *and* $r_1 \neq r_2$ *are called* hierarchy rules*.*
- *rules of the form* $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y)$ *with* $r_1, r_2, r_3 \in \boldsymbol{R}$ *and* $r_1 \neq r_2 \neq r_3$ *are called* intersection rules*.*
- *rules of the form* $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow \bot$ *with* $r_1, r_2 \in \boldsymbol{R}$ *and* $r_1 \neq r_2$ *are called* mutual exclusion rules*.*

Based on these definitions, we will prove Theorem 5.1.1, i.e., that SpeedE captures the core inference rules exactly and exclusively. To prove the theorem, we refer to the relevant propositions of ExpressivE proven in Chapter 4 and adapt them to SpeedE. For each of them, we give proofs, which in some situations follow from the ones in Chapter 4, and in other situations are entirely new constructions.

The key change of SpeedE that will be of our concern in the following proofs is fixing the width to a constant value, as this will require new proofs for some of the properties. Observe that SpeedE additionally changes the distance function of ExpressivE. However, this does not affect ExpressivE's inference capabilities, i.e., which inference rules can be captured. Careful inspection of the proofs of ExpressivE's inference capabilities given in Chapter 4 shows that the only property required of the distance function is that scores within the hyper-parallelogram are larger than those outside. As the newly defined distance function of SpeedE keeps this property, the change of distance function between the two models does not affect the proofs of the inference capabilities given in Chapter 4. Hence, the same proof argument can be applied.

The other observation that we will make in general before giving the specific proofs is that the "exactly" part, proven for ExpressivE in Propositions 4.2.9–4.2.15, of "exactly and exclusively" capturing rules is not affected by the changes in the SpeedE model. These proofs are all based on embedding pairs of entities as points in the virtual triple space and relations as hyper-parallelograms, which is still the case in SpeedE. Thus, we now proceed to prove that SpeedE captures the core inference rules exactly and exclusively.

Proposition 5.1.3 (Inversion (Exactly and Exclusively)). *Let* $\boldsymbol{m_h^+} = (\boldsymbol{M^+}, \boldsymbol{f_h^+})$ *be a relation configuration and* $r_1, r_2 \in \boldsymbol{R}$ *be relations where* $r_1(X,Y) \Leftrightarrow r_2(Y,X)$ *holds* *for any entities $X, Y \in \boldsymbol{E}$. Then $\boldsymbol{m}_h^+$ can capture $r_1(X,Y) \Leftrightarrow r_2(Y,X)$ exactly and exclusively.*

Proof. The proof of this property in Proposition 4.2.20 is based on a key assumption, namely that there is an $\boldsymbol{m_h}$ such that $\boldsymbol{f_h(r_1)}$ is the mirror image of $\boldsymbol{f_h(r_2)}$ with $\boldsymbol{f_h(r_1)} \neq \boldsymbol{f_h(r_2)}$. This is straightforward in ExpressivE but more complex in SpeedE. We will show this next.

Let us first observe that in SpeedE, it is not trivially given that there is an $\boldsymbol{m}_h^+ = (\boldsymbol{M}^+, \boldsymbol{f}_h^+)$ such that $\boldsymbol{f}_h^+\boldsymbol{(r_1)}$ is the mirror image of $\boldsymbol{f}_h^+\boldsymbol{(r_2)}$ with $\boldsymbol{f}_h^+\boldsymbol{(r_1)} \neq \boldsymbol{f}_h^+\boldsymbol{(r_2)}$, as $\boldsymbol{f_h(r_i)}$'s width embedding $\boldsymbol{w_i}$ has been replaced by a shared width constant w in $\boldsymbol{f}_h^+\boldsymbol{(r_i)}$ with $j \in \{1,2\}$. Thus, what needs to be shown is that there is a relation configuration $\boldsymbol{m}_h^+$ such that $\boldsymbol{f}_h^+\boldsymbol{(r_1)}$ is the mirror image of $\boldsymbol{f}_h^+\boldsymbol{(r_2)}$ with $\boldsymbol{f}_h^+\boldsymbol{(r_1)} \neq \boldsymbol{f}_h^+\boldsymbol{(r_2)}$, as then the original proof of ExpressivE can be directly applied to prove Proposition 5.1.3's claim, i.e., that $\boldsymbol{m}_h^+$ can capture $r_1(X,Y) \Leftrightarrow r_2(Y,X)$ exactly and exclusively. Now, it is interesting to see that fixing the width parameter in SpeedE as opposed to ExpressivE not only changes the model but actually allows a quite elegant construction witnessing this property.

Let us now give this construction, thereby showing the claim. Specifically, let $\boldsymbol{f}_h^+\boldsymbol{(r_1)} = (\boldsymbol{c_1}, w, \boldsymbol{s_1})$ with $\boldsymbol{c_1} = (\boldsymbol{c}_1^h || \boldsymbol{c}_1^t) \in \mathbb{R}^{2d}$, $w \in \mathbb{R}_{>0}$, and $\boldsymbol{s_1} = (\boldsymbol{s}_1^t || \boldsymbol{s}_1^h) \in \mathbb{R}^{2d}$. Furthermore, let $\boldsymbol{f}_h^+\boldsymbol{(r_2)} = (\boldsymbol{c_2}, w, \boldsymbol{s_2})$ with $\boldsymbol{c_2} = (\boldsymbol{c}_1^t || \boldsymbol{c}_1^h) \in \mathbb{R}^{2d}$, $w \in \mathbb{R}_{>0}$, and $\boldsymbol{s_2} = (\boldsymbol{s}_1^h || \boldsymbol{s}_1^t) \in \mathbb{R}^{2d}$. We will, in the following, show that the constructed $\boldsymbol{f}_h^+\boldsymbol{(r_2)}$ is the mirror image of $\boldsymbol{f}_h^+\boldsymbol{(r_1)}$ to prove our claim. Let $X, Y \in \boldsymbol{E}$ be arbitrary entities and let $\boldsymbol{f_v}$ be an arbitrary virtual assignment function defined over (X,Y) and (Y,X) with $\boldsymbol{f_v}(X,Y) = \boldsymbol{e_{xy}}$ and $\boldsymbol{f_v}(Y,X) = \boldsymbol{e_{yx}}$. Then by Inequality 5.2, a triple $r_1(X,Y)$ is captured to be true by $\boldsymbol{m}^+ = (\boldsymbol{M}^+, \boldsymbol{f}_h^+, \boldsymbol{f_v})$ if Inequality 5.3 is satisfied.

$$(\boldsymbol{e_{xy}} - \boldsymbol{c_1} - \boldsymbol{s_1} \odot \boldsymbol{e_{yx}})^{|.|} \preceq w \tag{5.3}$$
$$(\boldsymbol{e_{xy}} - (\boldsymbol{c}_1^h || \boldsymbol{c}_1^t) - (\boldsymbol{s}_1^t || \boldsymbol{s}_1^h) \odot \boldsymbol{e_{yx}})^{|.|} \preceq w \tag{5.4}$$
$$(\boldsymbol{e_{yx}} - (\boldsymbol{c}_1^t || \boldsymbol{c}_1^h) - (\boldsymbol{s}_1^h || \boldsymbol{s}_1^t) \odot \boldsymbol{e_{xy}})^{|.|} \preceq w \tag{5.5}$$
$$(\boldsymbol{e_{yx}} - \boldsymbol{c_2} - \boldsymbol{s_2} \odot \boldsymbol{e_{xy}})^{|.|} \preceq w \tag{5.6}$$

First, we replace $\boldsymbol{c_1}$ and $\boldsymbol{s_1}$ with their definitions based on $\boldsymbol{c}_1^h, \boldsymbol{c}_1^t, \boldsymbol{s}_1^h$, and $\boldsymbol{s}_1^t$, retrieving Inequality 5.4. Since Inequality 5.4 is element-wise, one can equivalently reformulate it by arbitrarily exchanging its dimensions. Using this insight, we can replace the head and tail dimensions for each embedding, thereby obtaining Inequality 5.5. Finally, by our construction of $\boldsymbol{f}_h^+(r_2)$, we have that $\boldsymbol{c_2} = (\boldsymbol{c}_1^t || \boldsymbol{c}_1^h)$ and $\boldsymbol{s_2} = (\boldsymbol{s}_1^h || \boldsymbol{s}_1^t)$. We substitute these equations into Inequality 5.5, thereby obtaining Inequality 5.6. Now, Inequality 5.6 states by the definition of a triple's truth (i.e., Inequality 5.2) that $r_2(Y,X)$ is captured by $\boldsymbol{m}_h^+$. Since Inequalities 5.3–5.6 are all equivalent, we have shown that $\boldsymbol{f}_h^+(r_1)$ is the mirror image of $\boldsymbol{f}_h^+(r_2)$. Since, it is now easy to see that an $\boldsymbol{m}_h^+$ exists such that $\boldsymbol{f}_h^+\boldsymbol{(r_1)}$

is the mirror image of $\boldsymbol{f_h^+(r_2)}$ with $\boldsymbol{f_h^+(r_1)} \neq \boldsymbol{f_h^+(r_2)}$, the proof of Proposition 4.2.20 can be directly applied to SpeedE. Thus, we have proven Proposition 5.1.3, i.e., that $\boldsymbol{m_h^+}$ can capture $r_1(X,Y) \Leftrightarrow r_2(Y,X)$ exactly and exclusively. □

	$\boldsymbol{c^h}$	$\boldsymbol{s^t}$	$\boldsymbol{c^t}$	$\boldsymbol{s^h}$
r_1	-2.5	0.5	1.5	0
r_2	1	-2	4.5	2

Table 5.1: Relation embeddings of a relation configuration $\boldsymbol{m_h^+}$ that captures hierarchy (i.e., $r_1(X,Y) \Rightarrow r_2(X,Y)$) exactly and exclusively using width $w = 1$.

Proposition 5.1.4 (Hierarchy (Exactly and Exclusively)). *Let $\boldsymbol{m_h^+} = (\boldsymbol{M^+}, \boldsymbol{f_h^+})$ be a relation configuration and $r_1, r_2 \in \boldsymbol{R}$ be relations where $r_1(X,Y) \Rightarrow r_2(X,Y)$ holds for any entities $X, Y \in \boldsymbol{E}$. Then $\boldsymbol{m_h^+}$ can capture $r_1(X,Y) \Rightarrow r_2(X,Y)$ exactly and exclusively.*

Proof. The proof of this property in Proposition 4.2.21 is based on the key assumption that there is an $\boldsymbol{m_h}$ such that $\boldsymbol{f_h(r_1)} \subset \boldsymbol{f_h(r_2)}$ with $\boldsymbol{f_h(r_1)} \neq \boldsymbol{f_h(r_2)}$. This is straightforward in ExpressivE but much more complex in SpeedE. We will show this next.

Let us first observe that in SpeedE, it is not trivially given that there is an $\boldsymbol{m_h^+} = (\boldsymbol{M^+}, \boldsymbol{f_h^+})$ such that $\boldsymbol{f_h^+(r_1)} \subset \boldsymbol{f_h^+(r_2)}$ with $\boldsymbol{f_h^+(r_1)} \neq \boldsymbol{f_h^+(r_2)}$, as $\boldsymbol{f_h(r_i)}$'s width embedding $\boldsymbol{w_i}$ has been replaced by a shared width constant w in $\boldsymbol{f_h^+(r_i)}$ with $j \in \{1,2\}$. Thus, what needs to be shown is that there is a relation configuration $\boldsymbol{m_h^+}$ such that $\boldsymbol{f_h^+(r_1)} \subset \boldsymbol{f_h^+(r_2)}$ with $\boldsymbol{f_h^+(r_1)} \neq \boldsymbol{f_h^+(r_2)}$, as then the original proof of ExpressivE can be directly applied to prove Proposition 5.1.4's claim, i.e., that $\boldsymbol{m_h^+}$ can capture $r_1(X,Y) \Rightarrow r_2(X,Y)$ exactly and exclusively. In the following, we construct such a relation configuration $\boldsymbol{m_h^+} = (\boldsymbol{M^+}, \boldsymbol{f_h^+})$, where $\boldsymbol{f_h^+(r_1)} \subset \boldsymbol{f_h^+(r_2)}$ with $\boldsymbol{f_h^+(r_1)} \neq \boldsymbol{f_h^+(r_2)}$ to prove the claim of Proposition 5.1.4:

Figure 5.1 (given in Section 5.1.1) visualizes the relation configuration $\boldsymbol{m_h^+} = (\boldsymbol{M^+}, \boldsymbol{f_h^+})$ provided in Table 5.1. As can be easily seen in Figure 5.1, $\boldsymbol{m_h^+}$ captures $\boldsymbol{f_h^+(r_1)} \subset \boldsymbol{f_h^+(r_2)}$ with $\boldsymbol{f_h^+(r_1)} \neq \boldsymbol{f_h^+(r_2)}$. Thus, we have proven Proposition 5.1.4, as (1) we have shown the existence of an $\boldsymbol{m_h^+}$ that captures $\boldsymbol{f_h^+(r_1)} \subset \boldsymbol{f_h^+(r_2)}$ with $\boldsymbol{f_h^+(r_1)} \neq \boldsymbol{f_h^+(r_2)}$ and (2) the proof of Proposition 4.2.21 can be directly applied to SpeedE since an $\boldsymbol{m_h^+}$ exists such that $\boldsymbol{f_h^+(r_1)} \subset \boldsymbol{f_h^+(r_2)}$ with $\boldsymbol{f_h^+(r_1)} \neq \boldsymbol{f_h^+(r_2)}$. □

Proposition 5.1.5 (Intersection (Exactly and Exclusively)). *Let $\boldsymbol{m_h^+} = (\boldsymbol{M^+}, \boldsymbol{f_h^+})$ be a relation configuration and $r_1, r_2, r_3 \in \boldsymbol{R}$ be relations where $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y)$ holds for any entities $X, Y \in \boldsymbol{E}$. Then $\boldsymbol{m_h^+}$ can capture $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y)$ exactly and exclusively.*

Proof Sketch. This is similar in construction to the previous proof. Hence, we only give a proof sketch for ease of readability. To prove Proposition 5.1.5, observe that in

	$\boldsymbol{c}^h$	$\boldsymbol{s}^t$	$\boldsymbol{c}^t$	$\boldsymbol{s}^h$
r_1	-3.75	0.5	1	0
r_2	1	-2	5	2
r_3	-3.5	0.5	0.5	-1

Table 5.2: Relation embeddings of a relation configuration $\boldsymbol{m}_h^+$ that captures intersection (i.e., $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y)$) exactly and exclusively using width $w = 1$.

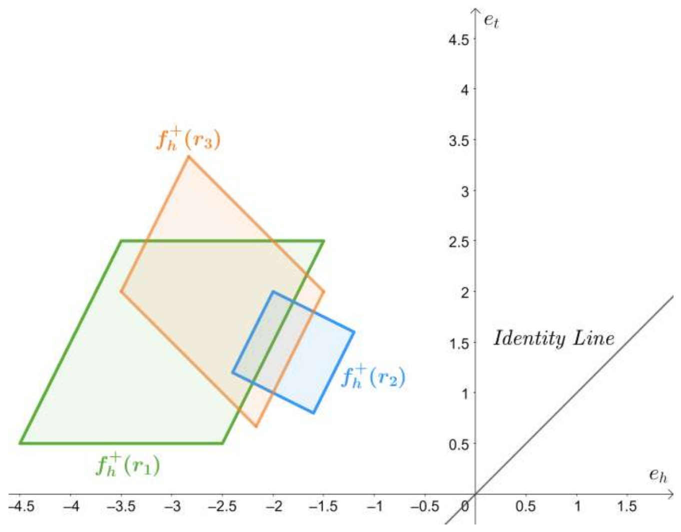

Figure 5.2: Relation embeddings of a relation configuration $\boldsymbol{m}_h^+$ that captures intersection (i.e., $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow r_3(X,Y)$) exactly and exclusively using width $w = 1$.

Proposition 4.2.22, an ExpressivE relation configuration $\boldsymbol{m}_h$ with several different width embeddings is constructed. However, the key observation we will make is that choosing the width embeddings differently is not necessary. In fact, an interested reader inspecting the original proof can obtain a proof applicable to SpeedE by following the proof of Proposition 4.2.22 analogously for the SpeedE relation configuration $\boldsymbol{m}_h^+$ described in Table 5.2 and visualized by Figure 5.2. Thus, the proof for Proposition 5.1.5 is straightforward given $\boldsymbol{m}_h^+$ defined in Table 5.2 and Proposition 4.2.22. □

Proposition 5.1.6 (**General Composition (Exactly and Exclusively)**)**.** *Let $r_1, r_2, r_3 \in \boldsymbol{R}$ be relations and let $\boldsymbol{m}_h^+ = (\boldsymbol{M}^+, \boldsymbol{f}_h^+)$ be a relation configuration, where $\boldsymbol{f}_h^+$ is defined over r_1, r_2, and r_3. Furthermore, let r_3 be the composite relation of r_1 and r_2, i.e., $r_1(X,Y) \wedge r_2(Y,Z) \Rightarrow r_3(X,Z)$ holds for all entities $X, Y, Z \in \boldsymbol{E}$. Then $\boldsymbol{m}_h^+$ can capture $r_1(X,Y) \wedge r_2(Y,Z) \Rightarrow r_3(X,Z)$ exactly and exclusively.*

Proof Sketch. This is similar in construction to the proof of Proposition 5.1.4. Hence, we only give a proof sketch for ease of readability. To prove Proposition 5.1.6, observe that in

	$\boldsymbol{c}^h$	$\boldsymbol{s}^t$	$\boldsymbol{c}^t$	$\boldsymbol{s}^h$
r_1	-7	3	5	1
r_2	-7.5	1	2	3
r_3	-19.5	2	13	2

Table 5.3: Relation embeddings of a relation configuration $\boldsymbol{m}_h^+$ that captures general composition (i.e., $r_1(X,Y) \wedge r_2(Y,Z) \Rightarrow r_3(X,Z)$) exactly and exclusively using width $w = 1$.

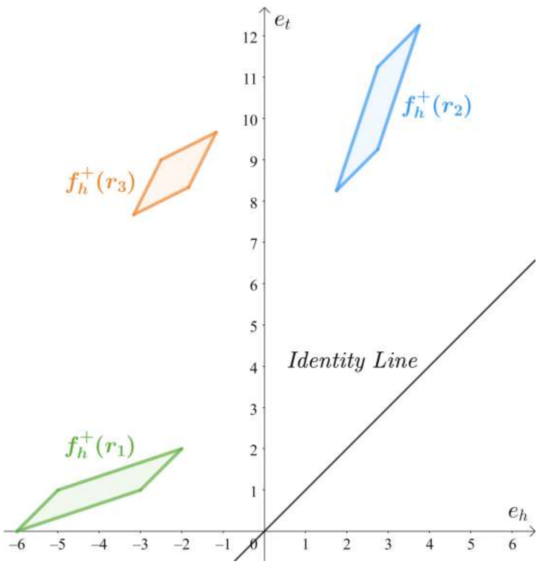

Figure 5.3: Relation embeddings of a relation configuration $\boldsymbol{m}_h^+$ that captures general composition (i.e., $r_1(X,Y) \wedge r_2(Y,Z) \Rightarrow r_3(X,Z)$) exactly and exclusively using width $w = 1$.

Proposition 4.2.23, an ExpressivE relation configuration $\boldsymbol{m}_h$ with several different width embeddings is constructed. However, choosing the width embeddings differently is not necessary. In fact, an interested reader inspecting the original proof can obtain a proof applicable to SpeedE by following the proof of Proposition 4.2.23 analogously for the SpeedE relation configuration $\boldsymbol{m}_h^+$ described in Table 5.3 and visualized by Figure 5.3. Thus, the proof for Proposition 5.1.6 is straightforward given $\boldsymbol{m}_h^+$ defined in Table 5.3 and Proposition 4.2.23. □

Proposition 5.1.7 (**Symmetry (Exactly and Exclusively)**). *Let $\boldsymbol{m}_h^+ = (\boldsymbol{M}^+, \boldsymbol{f}_h^+)$ be a relation configuration and $r_1 \in \boldsymbol{R}$ be a symmetric relation, i.e., $r_1(X,Y) \Rightarrow r_1(Y,X)$ holds for any entities $X, Y \in \boldsymbol{E}$. Then $\boldsymbol{m}_h^+$ can capture $r_1(X,Y) \Rightarrow r_1(Y,X)$ exactly and exclusively.*

Proposition 5.1.8 (**Anti-Symmetry (Exactly and Exclusively)**). *Let* $\boldsymbol{m}_{\boldsymbol{h}}^{+} = (\boldsymbol{M}^{+}, \boldsymbol{f}_{\boldsymbol{h}}^{+})$ *be a relation configuration and* $r_1 \in \boldsymbol{R}$ *be an anti-symmetric relation, i.e.,* $r_1(X,Y) \wedge r_1(Y,X) \Rightarrow \perp$ *holds for any entities* $X, Y \in \boldsymbol{E}$. *Then* $\boldsymbol{m}_{\boldsymbol{h}}^{+}$ *can capture* $r_1(X,Y) \wedge r_1(Y,X) \Rightarrow \perp$ *exactly and exclusively.*

The proofs for Proposition 5.1.7–5.1.8 are straightforward and work analogously to the proofs of Propositions 4.2.18 and 4.2.19. This is the case, as (1) any of these rules contain at most one relation, (2) thus we solely need to show that no unwanted rules over at most one relation are captured, as any considered rule over more than one relation (precisely inversion, hierarchy, intersection, and general composition) requires by Definition 5.1.2 at least two or three *distinct* relations and thus is not applicable, and (3) it is easy to see that, for instance, a relation hyper-parallelogram can be symmetric without being anti-symmetric, or vice versa (i.e., without capturing any unwanted rule).

Proposition 5.1.9 (**Mutual Exclusion (Exactly and Exclusively)**). *Let* $\boldsymbol{m}_{\boldsymbol{h}}^{+} = (\boldsymbol{M}^{+}, \boldsymbol{f}_{\boldsymbol{h}}^{+})$ *be a relation configuration and* $r_1, r_2 \in \boldsymbol{R}$ *be mutually exclusive relations, i.e.,* $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow \perp$ *holds for any entities* $X, Y \in \boldsymbol{E}$. *Then* $\boldsymbol{m}_{\boldsymbol{h}}^{+}$ *can capture* $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow \perp$ *exactly and exclusively.*

The proof for Proposition 5.1.9 is trivial, as it is straightforward to see that (1) there is an $\boldsymbol{m}_{\boldsymbol{h}}^{+} = (\boldsymbol{M}^{+}, \boldsymbol{f}_{\boldsymbol{h}}^{+})$ such that $\boldsymbol{f}_{\boldsymbol{h}}^{+}(r_1) \cap \boldsymbol{f}_{\boldsymbol{h}}^{+}(r_2) = \emptyset$, thereby $\boldsymbol{m}_{\boldsymbol{h}}^{+}$ captures $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow \perp$ exactly, (2) neither $\boldsymbol{f}_{\boldsymbol{h}}^{+}(r_1)$ nor $\boldsymbol{f}_{\boldsymbol{h}}^{+}(r_2)$ need to be symmetric, thereby no unwanted symmetry rule is captured, (3) $\boldsymbol{f}_{\boldsymbol{h}}^{+}(r_1)$ does not need to be the mirror image of $\boldsymbol{f}_{\boldsymbol{h}}^{+}(r_2)$, thus no unwanted inversion rule is captured, and finally (4) since $\boldsymbol{f}_{\boldsymbol{h}}^{+}(r_1)$ and $\boldsymbol{f}_{\boldsymbol{h}}^{+}(r_2)$ are disjoint, neither $\boldsymbol{f}_{\boldsymbol{h}}^{+}(r_1)$ can subsume $\boldsymbol{f}_{\boldsymbol{h}}^{+}(r_2)$ nor vice versa, thus no unwanted hierarchy rule is captured. Thus by Points 1–4, we have shown that $\boldsymbol{m}_{\boldsymbol{h}}^{+}$ captures $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow \perp$ exactly and that it does not capture any unwanted positive rule that is applicable, i.e., requires at most two different relations (symmetry, inversion, and hierarchy). Thus, we have shown Proposition 5.1.9, i.e., that $\boldsymbol{m}_{\boldsymbol{h}}^{+}$ can capture $r_1(X,Y) \wedge r_2(X,Y) \Rightarrow \perp$ exactly and exclusively.

Finally, by Propositions 5.1.3–5.1.9, we have shown Theorem 5.1.1, i.e., that SpeedE captures all core inference rules exactly and exclusively.

5.2 Experimental Setup Details

This section discusses our experiment design to establish the basis for the upcoming empirical results. In particular, this section presents the concrete experimental setup, including details of the implementation, used hardware, learning setup, chosen hyperparameters, and CO_2 emissions.

Implementation Details. As for ExpressivE in Chapter 4, we have implemented our gKGE using PyKEEN 1.7 (Ali et al., 2021), a Python library that runs under the MIT license and offers support for numerous benchmarks and gKGEs. In doing so, we facilitate the comfortable reuse of SpeedE for upcoming benchmarks and applications. To ease reproducing our findings, we provide SpeedE's source code in a public GitHub repository[1] together with a ReadMe.md file stating library dependencies and running instructions.

Training Setup. Following our approach in Chapter 4, we train SpeedE and ExpressivE for up to 1000 epochs using gradient descent and the Adam optimizer (Kingma and Ba, 2015) and stop the training if the validation H@10 score does not increase by minimally 0.5% for WN18RR, YAGO3-10, and 1% for FB15k-237 after 100 epochs. We average the experimental results over three runs on each benchmark to handle marginal performance fluctuations. We have trained each model on one of four GeForce RTX 2080 Ti GPUs of our internal cluster. In particular, during the training phase, we optimize the self-adversarial negative sampling loss (Sun et al., 2019) using the Adam optimizer (Kingma and Ba, 2015). Following our approach in Chapter 4, we train SpeedE and ExpressivE for up to 1000 epochs using gradient descent and the Adam optimizer (Kingma and Ba, 2015) and stop the training early if after 100 epochs the validation H@10 score does not rise by minimally 0.5% for WN18RR and YAGO3-10, and 1% for FB15k-237. Following the procedure of Chami et al. (2020), we employ the standard augmentation protocol of Lacroix et al. (2018), adding inverse relations to the benchmarks. Any experiment was run three times to average over light performance variations. We will discuss the optimization of hyperparameters in the following paragraph.

Hyperparameter Optimization. Following similar optimization principles as Balazevic et al. (2019a); Chami et al. (2020), and our approach in Chapter 4, we manually tuned the following hyperparameters within the listed ranges: (1) the learning rate $\lambda \in \{b*10^{-c} \mid b \in \{1,2,5\} \wedge c \in \{2,3,4,5,6\}\}$, (2) the negative sample size $n \in \{100,150,200,250\}$, (3) the loss margin $\gamma \in \{2,3,4,5,6\}$, (4) the adversarial temperature $\alpha \in \{1,2,3,4\}$, (5) the batch size $b \in \{100,250,500,1000,2000\}$, and (6) constraining the distance slope parameters to be equal — i.e., $s_i^t = s_i^o$ for each relation $r_i \in \boldsymbol{R}$ — or not $EqDS \in \{true, false\}$. Following the literature (Chami et al., 2020; Lu and Hu, 2020), we used for the large YAGO3-10 benchmark a wider range for the negative sampling size n, in particular $n \in \{100,200,500,1000,2000\}$. Similar to Lu and Hu (2020), we also increased the range for margins γ to include 50 and 100 for YAGO3-10. In accordance with our approach in Chapter 4, we chose self-adversarial negative sampling (Sun et al., 2019) for generating negative triples. We list the best hyperparameters for SpeedE split by benchmark and embedding dimensionality in Table 5.4. Following Chami et al. (2020), we used one parameter set for any low-dimensional experiment (i.e., $d \leq 50$) and one parameter set for any high-dimensional experiment (i.e., $d > 50$). Furthermore, for ExpressivE, we used the hyperparameters of Table 4.4 (in Chapter 4) under high-dimensional conditions, as they correspond to the best-published results for ExpressivE (Pavlović and Sallinger, 2023b). For low-dimensional conditions, ExpressivE's best hyperparameter setting was unknown. Thus, we optimized ExpressivE's hyperparameters manually, finding the hyperparameters of Table 5.5 to produce the best KGC results for ExpressivE under low dimensionalities. For RotH, we used the hyperparameters of Chami et al. (2020), as they report the

[1] https://github.com/AleksVap/SpeedE

best-published results for RotH. Finally, we used the same hyperparameters for each of SpeedE's model variants (Min_SpeedE, Diff_SpeedE, and Eq_SpeedE, which are introduced in Sections 5.1 and 5.3) to compare SpeedE to them directly.

Dataset	Embedding Dimensionality	Margin	Learning Rate	Adversarial Temperature	Negative Sample Size	Batch Size	EqDS
WN18RR	$d \leq 50$	3	$5 * 10^{-3}$	2	200	250	false
WN18RR	$d > 50$	3	$1 * 10^{-3}$	2	200	250	true
FB15k-237	$d \leq 50$	2	$5 * 10^{-4}$	4	250	100	false
FB15k-237	$d > 50$	4	$1 * 10^{-4}$	4	150	1000	false
YAGO3-10	$d \leq 50$	100	$1 * 10^{-2}$	2	2000	2000	false

Table 5.4: Hyperparameters of SpeedE models that achieve the best performance on WN18RR, FB15k-237, and YAGO3-10 split by low-dimensional (i.e., $d \leq 50$) and high-dimensional setting (i.e., $d > 50$).

Dataset	Embedding Dimensionality	Margin	Learning Rate	Adversarial Temperature	Negative Sample Size	Batch Size
WN18RR	$d \leq 50$	2	$5 * 10^{-3}$	3	200	250
FB15k-237	$d \leq 50$	2	$5 * 10^{-4}$	4	250	100
YAGO3-10	$d \leq 50$	100	$1 * 10^{-2}$	2	2000	2000

Table 5.5: Hyperparameters of ExpressivE that achieve the best performance on WN18RR, FB15k-237, and YAGO3-10 under low-dimensional conditions (i.e., $d \leq 50$).

Evaluation. In all of our experiments we: (*i*) employ the standard benchmarks, (*ii*) follow the standard evaluation protocol, and (*iii*) presents the standard metrics (filtered MRR and H@k) for KGC. Each of these parts is described in detail in Section 2.1.1.

CO_2 Emissions. The sum of all reported experiments took less than 150 GPU hours. This corresponds to an estimate of approximately $16.20kg\ CO_2\text{-}eq$, based on the OECD's 2014 carbon efficiency average of $0.432kg/kWh$ and the usage of an RTX 2080 Ti on private infrastructure. We computed these estimates using the Machine Learning Emissions Calculator (Lacoste et al., 2019).

5.3 Experiments

This section empirically evaluates SpeedE. Section 5.3.1 briefly describes the selected benchmarks and baseline models. Section 5.3.2 studies SpeedE's KGC performance, finding that it is competitive with state-of-the-art gKGEs on FB15k-237 and even significantly outperforms them on YAGO3-10 and WN18RR. Section 5.3.3 studies SpeedE's space and time efficiency, finding that on WN18RR, SpeedE needs a quarter of ExpressivE's parameters to reach the same KGC performance while training five times faster than it.

5.3.1 Benchmarks and Baselines

Dataset	$\|\boldsymbol{E}\|$	$\|\boldsymbol{R}\|$	C_G	κ
FB15k-237	14,541	237	-0.65	(1.00, 0.18, 0.36, 0.06)
WN18RR	40,943	11	-2.54	(1.00, 0.61, 0.99, 0.50)
YAGO3-10	123,143	37	-0.54	-

Table 5.6: Benchmark dataset characteristics. Curvature C_G is from (Chami et al., 2020); the lower, the more hierarchical the data. Krackhardt scores κ are from (Bai et al., 2021); the higher, the more hierarchical the data.

Datasets. We empirically evaluate SpeedE on the three standard KGC benchmarks, WN18RR (Bordes et al., 2013; Dettmers et al., 2018), FB15k-237 (Bordes et al., 2013; Toutanova and Chen, 2015), and YAGO3-10 (Mahdisoltani et al., 2015) (for detailed information see Section 2.1.1).

Characteristics. Table 5.6 displays the following characteristics of the benchmarks: their number of entities $|\boldsymbol{E}|$ and relations $|\boldsymbol{R}|$, their curvature C_G (taken from Chami et al. (2020)), and the Krackhardt scores κ (taken from Bai et al. (2021)), consisting of the four metrics: (*connectedness*, *hierarchy*, *efficiency*, *LUBedness*). Both C_G and κ state how tree-like a benchmark is and, thus, how hierarchical its relations are.

Baselines. We compare our SpeedE model to (1) the Euclidean gKGEs ExpressivE (introduced in Chapter 4), HAKE (Zhang et al., 2020), TuckER (Balazevic et al., 2019b), MuRE (Balazevic et al., 2019a), and RefE, RotE, and AttE (Chami et al., 2020), (2) the complex gKGEs ComplEx-N3 (Lacroix et al., 2018) and RotatE (Sun et al., 2019), and (3) the hyperbolic gKGEs ConE (Bai et al., 2021), MuRP (Balazevic et al., 2019a), and RefH, RotH, and AttH (Chami et al., 2020). Furthermore, as in (Chami et al., 2020), we evaluate SpeedE and ExpressivE in the low-dimensional setting using an embedding dimensionality of 32.

5.3.2 Knowledge Graph Completion

In this section, we evaluate the KGC performance of SpeedE and state-of-the-art gKGEs. Next, we study how well these models represent hierarchical relations, on which hyperbolic gKGEs are typically most effective (Balazevic et al., 2019a; Chami et al., 2020). Finally, we analyze the effect of the embedding dimensionality on SpeedE's KGC performance and perform an ablation study to understand the necessity of its distance slope parameters.

Low-Dimensional KGC. Following the evaluation protocol of Chami et al. (2020), we evaluate each gKGE's performance under $d = 32$. We report the MRR and H@k for $k \in \{1, 3, 10\}$ in Table 5.7. The table reveals that on YAGO3-10 — the largest benchmark, containing over a million triples — SpeedE outperforms any state-of-the-art gKGE by a relative difference of 7% on H@1, providing strong evidence for SpeedE's scalability to large KGs. Furthermore, it shows that our enhanced SpeedE model is competitive with state-of-the-art gKGEs on FB15k-237 and even outperforms any competing gKGE on WN18RR by a large margin. Furthermore, SpeedE's performance gain over Min_SpeedE on the highly hierarchical dataset WN18RR (see Table 5.6) provides strong empirical evidence for the effectiveness of the distance slope parameters for representing hierarchical relations under low-dimensional conditions. SpeedE's performance on the more hierarchical WN18RR already questions the necessity of hyperbolic gKGEs for representing hierarchical relations, which will be further investigated in the following.

Space	Model	**WN18RR**				**FB15k-237**				**YAGO3-10**			
		MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
Euclidean	SpeedE	**.493**	**.446**	**.512**	**.584**	**.320**	**.227**	**.356**	**.504**	**.413**	**.332**	**.453**	**.564**
	Min_SpeedE	.485	.442	.499	.573	.319	.226	**.356**	.502	.410	.328	.449	.563
	ExpressivE	.485	.442	.499	.571	.298	.208	.331	.476	.333	.257	.367	.476
	TuckER	.428	.401	-	.474	.306	.223	-	.475	-	-	-	-
	MuRE	.458	.421	.471	.525	.313	.226	.340	.489	.283	.187	.317	.478
	RefE	.455	.419	.470	.521	.302	.216	.330	.474	.370	.289	.403	.527
	RotE	.463	.426	.477	.529	.307	.220	.337	.482	.381	.295	.417	.548
	AttE	.456	.419	.471	.526	.311	.223	.339	.488	.374	.290	.410	.537
	HAKE	.416	.389	.427	.467	.296	.212	.323	.463	.253	.164	.286	.430
Non-Euclidean	RotatE	.387	.330	.417	.491	.290	.208	.316	.458	.235	.153	.260	.410
	ComplEx-N3	.420	.390	.420	.460	.294	.211	.322	.463	.336	.259	.367	.484
	MuRP	.465	.420	.484	.544	.323	.235	.353	**.501**	.230	.150	.247	.392
	RefH	.447	.408	.464	.518	.312	.224	.342	.489	.381	.302	.415	.530
	RotH	**.472**	.428	**.490**	**.553**	.314	.223	.346	.497	.393	.307	.435	.559
	AttH	.466	.419	.484	.551	**.324**	**.236**	**.354**	**.501**	**.397**	**.310**	**.437**	**.566**
	ConE	.471	**.436**	.486	.537	-	-	-	-	-	-	-	-

Table 5.7: Low-dimensional ($d = 32$) KGC performance of SpeedE, Min_SpeedE, ExpressivE, and state-of-the-art gKGEs on WN18RR, FB15k-237, and YAGO3-10 split by embedding space. The results of SpeedE, Min_SpeedE, and ExpressivE were obtained by us; ConE are from (Bai et al., 2021); HAKE and RotatE are from (Zheng et al., 2022); TuckER are from (Wang et al., 2021); and any other gKGE are from (Chami et al., 2020).

Hierarchical Relations (Chami et al., 2020; Zhang et al., 2020) describe hierarchies between entities, such as *part_of*. Hyperbolic gKGEs have shown great potential for representing hierarchical relations, outperforming Euclidean gKGEs under low-dimensional conditions, thereby justifying the increased model complexity added by the hyperbolic space (Chami et al., 2020). To study SpeedE's performance on hierarchical relations, we evaluate SpeedE on the triples of any hierarchical relation of WN18RR following the methodology of Bai et al. (2021). Table 5.8 presents the results of this study. It reveals that SpeedE significantly improves over ExpressivE on most relations and outperforms RotH on five out of the seven hierarchical ones. Most notably, SpeedE improves over RotH by a relative difference of 23% on H@10 on the hierarchical relation *_member_of_domain_usage*, providing empirical evidence for SpeedE's promising potential for representing hierarchical relations even under low-dimensional settings. The performance gain on hierarchical relations is likely due to the added distance slope parameters, which allow for independently adjusting the distance function's slopes.

Relation	ExpressivE	RotH	SpeedE
_member_meronym	0.362	**0.399**	0.379
_hypernym	0.276	0.276	**0.301**
_has_part	0.308	**0.346**	0.330
_instance_hypernym	0.509	0.520	**0.543**
_member_of_domain_region	0.365	0.365	**0.397**
_member_of_domain_usage	**0.545**	0.438	0.538
_synset_domain_topic_of	0.468	0.447	**0.502**

Table 5.8: H@10 of ExpressivE, RotH, and SpeedE on hierarchical relations (Bai et al., 2021) of WN18RR.

Dimensionality Study. To analyze the effect of the embedding dimensionality on the KGC performance, we evaluate state-of-the-art gKGEs on WN18RR under varied dimensionalities. Figure 5.4 visualizes the results of this study, displaying error bars for our SpeedE model with average MRR and standard deviation computed over three runs. The figure reveals that, surprisingly, ExpressivE significantly outperforms RotH, especially under low-dimensional conditions, and that the enhanced SpeedE model achieves an additional performance improvement over ExpressivE. This result provides further evidence for the great potential of Euclidean gKGEs under low-dimensional conditions.

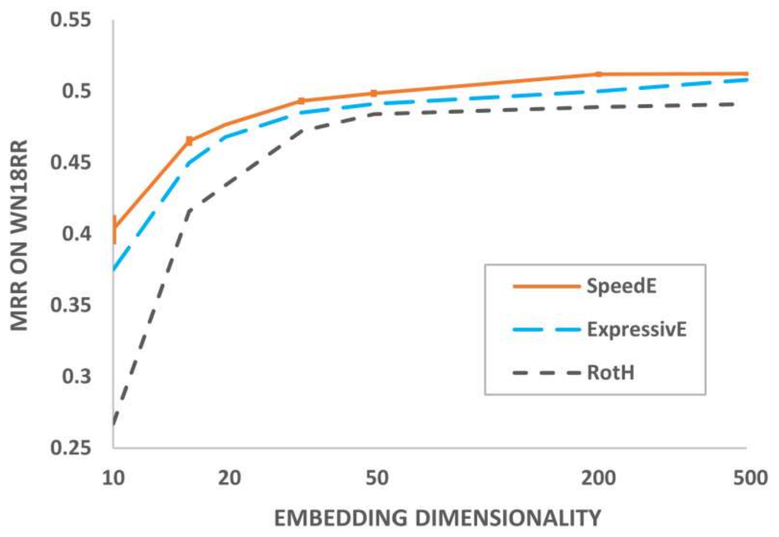

Figure 5.4: MRR of the best gKGEs on WN18RR with $d \in \{10, 16, 20, 32, 50, 200, 500\}$.

High-Dimensional KGC. Table 5.9 displays the KGC performance of state-of-the-art gKGEs under high-dimensional conditions (i.e., $d \geq 200$), where the results for SpeedE were obtained by us, for ExpressivE are from Table 4.6 (in Chapter 4), for HAKE are from (Zhang et al., 2020), for ConE are from (Bai et al., 2021), for BoxE are from (Abboud et al., 2020), for MuRE and MuRP are from (Balazevic et al., 2019a; Chami et al., 2020), for DistMult are from (Dettmers et al., 2018), for RotatE are from (Sun et al., 2019), for TuckER are from (Balazevic et al., 2019b), and for any other gKGE are from (Chami et al., 2020). Table 5.9 reveals that on FB15k-237, SpeedE achieves highly competitive KGC performance compared to gKGEs of its own family while dramatically outperforming any competing gKGE on WN18RR. Note that for sustainability reasons and due to limited computational resources, the high-dimensional setting for the largest benchmark, YAGO3-10, was not performed.

Family	Model	**WN18RR**				**FB15k-237**			
		MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
Functional / Spatial	SpeedE	**.512**	$\underline{.460}$	.531	**.615**	$\underline{.348}$	$\underline{.253}$	0.386	.536
	ExpressivE	.508	**.464**	.522	.597	**.350**	**.256**	.387	.535
	HAKE	.497	.452	.516	.582	.346	.250	.381	**.542**
	ConE	.496	.453	.515	.579	.345	.247	.381	.540
	BoxE	.451	.400	.472	.541	.337	.238	.374	.538
	MuRE	.475	.436	.487	.554	.336	.245	.370	.521
	RefE	.473	.430	.485	.561	.351	.256	.390	$\underline{.541}$
	RotE	.494	.446	.512	.585	.346	.251	.381	.538
	AttE	.490	.443	.508	.581	.351	.255	.386	.543
	MuRP	.481	.440	.495	.566	.335	.243	.367	.518
	RefH	.461	.404	.485	.568	.346	.252	.383	.536
	RotH	.496	.449	.514	.586	.344	.246	.380	.535
	AttH	.486	.443	.499	.573	$\underline{.348}$	.252	.384	.540
Bilinear	DistMult	.430	.390	.440	.490	.241	.155	.263	.419
	RotatE	.476	.428	.492	.571	.338	.241	.375	.533
	ComplEx-N3	.480	.435	.495	.572	.357	.264	.392	.547
	QuatE	**.488**	.438	.508	**.582**	.348	.248	.382	**.550**
	TuckER	.470	**.443**	.482	.526	**.358**	**.266**	.394	.544

Table 5.9: KGC performance under high dimensionalities of SpeedE and state-of-the-art gKGEs on WN18RR and FB15k-237 split by model family.

Ablation Study. Finally, to study the necessity of s_i^t and s_i^o in SpeedE, we introduce two versions of SpeedE: (1) Eq_SpeedE that forces $s_i^t = s_i^o$ and (2) Diff_SpeedE, where s_i^t and s_i^o can be different. We hypothesize that the flexibility of different s_i^t and s_i^o might be beneficial under lower dimensionalities, while under higher dimensionalities, reducing the number of parameters and thus setting $s_i^t = s_i^o$ might be beneficial. Figure 5.5 visualizes the result of this analysis, using embedding dimensionalities of $d \in \{10, 16, 20, 32, 50, 200, 500\}$. It empirically supports our hypothesis, as Diff_SpeedE outperforms Eq_SpeedE under low dimensionalities and vice-versa in high ones.

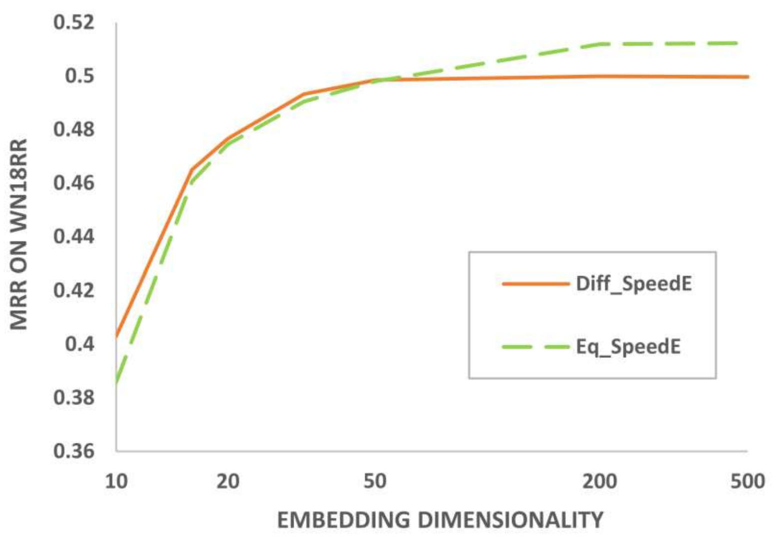

Figure 5.5: MRR of different ablations of SpeedE on WN18RR

5.3.3 Space and Time Efficiency

This section empirically analyzes SpeedE's space and time efficiency compared to state-of-the-art gKGEs.

Time per Epoch. Following the methodology of Wang et al. (2021), Table 5.10 displays the training time per epoch of SpeedE and state-of-the-art gKGEs for WN18RR, FB15k-237, and YAGO3-10 with dimensionality $d = 32$, negative sampling size $n = 500$, and batch size $b = 500$. The times per epoch were recorded on a GeForce RTX 2080 Ti GPU of our internal cluster. The empirical results of the table align with the theoretical results of Sections 5.1.1 and 5.1.2, stating that SpeedE approximately halves ExpressivE's inference time and, thus, also its time per epoch. Moreover, the results emphasize the efficiency benefits of SpeedE over state-of-the-art gKGEs, revealing that under the same configurations, SpeedE solely requires about a sixth of RotH's and AttH's time per epoch.

To provide a fair space and time efficiency comparison, we measure the convergence time of gKGEs with roughly equal KGC performance. Specifically, we observe that SpeedE with dimensionality $d = 50$ achieves comparable or slightly better KGC performance on WN18RR to ExpressivE with $d = 200$ and the best-published results of RotH, HAKE, and ConE with $d = 500$. In particular, the results are summarized in Table 5.11.

Hypotheses. Since (1) the dimensionality of SpeedE embeddings is much smaller in comparison to RotH's, HAKE's, ConE's, and ExpressivE's dimensionality, while (2) SpeedE achieves comparable or even slightly better KGC performance, we expect a considerable improvement in SpeedE's space and time efficiency at comparable KGC performance. Next, based on Table 5.11's results, we analyze how strongly SpeedE reduces the model size and convergence time of competing gKGEs.

Model	**Time per Epoch**		
	WN18RR	FB15k-237	YAGO3-10
SpeedE	7s	22s	88s
ExpressivE	15s	46s	185s
RotH	42s	112s	520s
AttH	43s	113s	533s

Table 5.10: Time per epoch of SpeedE, ExpressivE, RotH, and AttH.

Model	Dim.	MRR	Conv. Time	#Parameters
SpeedE	**50**	**.500**	**6min**	**2M**
ExpressivE	200	**.500**	31min	8M
HAKE	500	.497	50min	41M
ConE	500	.496	1.5h	20M
RotH	500	.496	2h	21M

Table 5.11: Dimensionality, MRR, convergence time, and number of parameters of state-of-the-art gKGE's on WN18RR.

Model Size Analysis. Since $|\boldsymbol{R}| << |\boldsymbol{E}|$ in most graphs, (WN18RR: $|\boldsymbol{R}|/|\boldsymbol{E}| = 0.00012$) and since SpeedE, ExpressivE, ConE, and RotH embed each entity with a single real-valued vector, SpeedE ($d = 50$) needs solely a quarter of ExpressivE's ($d = 200$) and a tenth of ConE's and RotH's ($d = 500$) number of parameters, while preserving their KGC performance on WN18RR (Table 5.11). As HAKE requires two real-valued vectors per entity, SpeedE ($d = 50$) solely needs a twentieth of HAKE's ($d = 500$) parameters to achieve a slightly better KGC performance. Table 5.11 lists the number of parameters of a trained SpeedE model and state-of-the-art gKGEs, empirically confirming that SpeedE significantly reduces the size of competing gKGEs.

Convergence Time Analysis. To quantify the convergence time, we measure for each gKGE the time to reach a validation MRR score of 0.490, i.e., approximately 1% less than the worst reported MRR score of Table 5.11. As outlined in the table, SpeedE converges already after $6min$. Thus, while keeping strong KGC performance on WN18RR, SpeedE speeds up ExpressivE's convergence time by a factor of 5, HAKE's by a factor of 9, ConE's by a factor of 15, and RotH's by a factor of 20.

Discussion. These results show that SpeedE is not only competitive with state-of-the-art gKGEs on FB15k-237 and significantly outperforms them on YAGO3-10 and WN18RR but even preserves their KGC performance on WN18RR with much fewer parameters and a dramatically shorter convergence time, in particular speeding up the convergence time of the ExpressivE model by a factor of 5, while using solely a fourth of its number of parameters.

5.4 Summary

Although there has been much work on resource-efficient gKGEs, any such work has focused exclusively on reducing the embedding dimensionality (Balazevic et al., 2019a; Chami et al., 2020; Bai et al., 2021) or using simpler embedding spaces (Kazemi and Poole, 2018; Zhang et al., 2020; Pavlović and Sallinger, 2023b), thus addressing only one side of the efficiency problem.

This chapter has reached our second research goal (G2) by jointly addressing the embedding space and dimensionality sides. In particular, we introduce SpeedE, a lightweight gKGE that — as required by G2 — (*i*) dramatically increases the efficiency of current gKGEs, needing solely a fifth of the training time and a fourth of the number of parameters of the state-of-the-art ExpressivE model on WN18RR to reach the same KGC performance. Even more, (*ii*) SpeedE is competitive with state-of-the-art gKGEs on FB15k-237 while even significantly outperforming them on WN18RR and YAGO3-10, the largest benchmark containing over a million triples. Therefore, our model overcomes the scalability divide, strengthening the bond between the ML, DB, and SW fields. Similarly to ExpressivE, we show that SpeedE also captures all core inference rules. Thus, SpeedE additionally even satisfies G1, bridging also the reasoning divide.

CHAPTER 6

Data Management Divide

The goal of this chapter is to reach G3 (see Chapter 1), i.e., to bridge the data management divide for KGs between the DB and SW communities by developing one uniform and consistent KG management system that satisfies the requirements of both communities. More specifically:

- **Theoretical Translation**. We provide a uniform and complete framework to integrate SPARQL support into a KG language that meets all of the DB and SW requirements R1–R5, described in Chapter 1. We have thus extended, simplified, and – in some cases – corrected previous approaches of translating SPARQL queries (under both set and bag semantics) to various Datalog dialects (Polleres, 2007; Polleres and Wallner, 2013; Angles and Gutierrez, 2016a). For instance, to the best of our knowledge, all previous translations have missed or did not consider correctly certain aspects of the SPARQL standard of the zero-or-one and zero-or-more property paths.
- **Translation Engine.** On top of the Vadalog system, we have developed SparqLog, a translation engine that covers most of the considered SPARQL 1.1 functionality. We thus had to fill several gaps between the abstract theory and the practical development of the translation engine. For instance, we have designed specific Skolem functions to support bag semantics and to generate a universal duplicate preservation process. On the other hand, using the Vadalog system as the basis of our engine made significant simplifications possible (such as letting Vadalog take care of complex filter constraints), and we also got ontological reasoning "for free". SparqLog, therefore, supports both query answering and ontological reasoning in a single uniform and consistent system.
- **Experimental Evaluation**. We carry out an extensive empirical evaluation on multiple benchmarks with two main goals in mind: to verify the compliance of

SparqLog with the SPARQL standard and to compare our system's performance with comparable ones. It turns out that while SparqLog covers a great part of the selected SPARQL 1.1 functionality correctly, some other systems (specifically Virtuoso) employ a non-standard behavior on queries containing property paths. As far as query-execution times are concerned, the performance of SparqLog is, in general, comparable to other systems such as the SPARQL system Fuseki or the querying and reasoning system Stardog, and it significantly outperforms these systems on complex queries containing recursive property paths and involving ontologies.

Organization. This chapter contains material from the following constituent papers of this dissertation (Angles et al., 2023a,b). Furthermore, note that this chapter, including its introduction, reuses content from my Bachelor's (Pavlović, 2019) and Master's theses (Pavlović, 2020). Thus, we make the reused and novel content for each of the following sections explicit in Section 6.1. Next, we present the general principles of our SparqLog system in Section 6.2, formally define our system's translation from SPARQL 1.1 to a suitable Datalog dialect in Section 6.3, and prove the correctness of this translation in Section 6.4. Moreover, in Section 6.5, we discuss some essential details on SparqLog's implementation. In Section 6.6, we identify suitable benchmarks and design experiments for (*i*) testing SparqLog's compliance to SPARQL and measuring its performance on (*ii*) query answering and (*iii*) ontological reasoning, subsequently presenting the retrieved results. Section 6.7 summarizes this chapter's key insights.

6.1 Details on SparqLog's Complete Presentation

This section extensively discusses which parts of this chapter (*i*) are taken from my Bachelor's (Pavlović, 2019) and Master's theses (Pavlović, 2020), solely being stated here for completeness, and (*ii*) constitute novel content, i.e., contributions of this dissertation.

Theoretical Translation. A first version of the translation from SPARQL 1.1 to the Datalog dialect, Warded Datalog$^{\pm}$, was already covered in (Pavlović, 2019, 2020). Thus, the SPARQL feature prioritization (Section 6.2.3) was largely taken from (Pavlović, 2020) (extended by additional supported features as discussed in the next paragraph). The translations of RDF graphs (Section 6.3.1) and SPARQL graph patterns (Section 6.3.2) are taken from (Pavlović, 2019). We slightly modify them for consistency with the rest of this work. Furthermore, the translation of property path patterns (Section 6.3.3) was already presented in the form of algorithms in (Pavlović, 2020). Section 6.3.3 reformulates these algorithms to formal definitions of a translation function for property path expressions, thereby ensuring consistency with the translation function of RDF graphs and SPARQL graph patterns presented in Sections 6.3.1 and 6.3.2. The translation of the SELECT query form (presented in Section 6.3.4) was taken from (Pavlović, 2019). At the same time, the translation of the ASK query form (also in Section 6.3.4) was added in this dissertation for completeness. A novel and vital contribution of this dissertation w.r.t. the presented theoretical translation are proofs for its correctness, covered in Section 6.4.

Implementation and Benchmarks. Also, a prototypical version of the SparqLog system was already implemented in the course of my Master's thesis (Pavlović, 2020). Thus, the implementation details (covered in Section 6.5) are taken from (Pavlović, 2020). Furthermore, the selection of appropriate benchmarks for the compliance of SparqLog to the SPARQL 1.1 standard (Section 6.6.1), as well as the corresponding compliance analysis (Section 6.6.2) are primarily taken from (Pavlović, 2020). This dissertation extends the benchmark analysis of (Pavlović, 2020) by adding a discussion on appropriate performance benchmarks for measuring query execution times in Section 6.6.1 (Paragraph Performance Benchmarking). Additionally, we extend the SparqLog system by missing SPARQL features in this dissertation, including (*i*) complex expressions in ORDER BY and COUNT statements, (*ii*) REGEX functions, such as UCASE and CONTAINS, (*iii*) the DATATYPE feature, and (*iv*) the "exactly n occurrences", "n or more occurrences", and "between 0 and n occurrences" property path features. Furthermore, we optimize SparqLog's implementation with standard query optimization techniques of the DB community by rewriting the query execution tree based on relational algebra, for instance, pushing selection operations toward the leaf nodes to convert cross-products to joins. These contributions of my dissertation — i.e., SparqLog's support of additional SPARQL features and enhanced query optimization — allow SparqLog to reach compliance with the SPARQL standard on all benchmark queries (see Section 6.6.2). In contrast, SparqLog's prototype (Pavlović, 2020) could not answer (*i*) 9 queries of the FEASIBLE benchmark due to unsupported SPARQL features and (*ii*) two queries of the SP2Bench benchmark due to time-outs.

Further Contributions of this Dissertation. A key contribution of this dissertation is the measurement of the query answering performance of SparqLog (presented in Section 6.6.3). In a nutshell, the measurements show that — compared to standard SPARQL querying systems, such as Virtuoso and Fuseki — SparqLog is not solely highly competitive with the very fast Virtuoso system on regular SPARQL queries but can even (correctly) answer many more property path queries. Furthermore, SparqLog dramatically outperforms Fuseki on query execution time while keeping its ability to follow the SPARQL standard accurately. Finally, Section 6.6.4 covers another essential contribution of this dissertation. Specifically, in Section 6.6.4, we extend SP2Bench with triples describing subproperty and subclass relationships, subsequently using this extended benchmark to evaluate SparqLog's ontological reasoning performance. We find that SparqLog is faster than the state-of-the-art reasoning system Stardog for most queries, especially for queries with recursive property paths that contain two variables.

Summary of Contributions. This chapter makes the following contributions:

- While the theoretical translation from SPARQL 1.1 to Warded Datalog$^{\pm}$ was taken from (Pavlović, 2019, 2020), it is (*i*) split across these works and (*ii*) presented as translation functions in one work (Pavlović, 2019) and as algorithms in the other (Pavlović, 2020). Sections 6.3.2–6.3.4 consolidate, rewrite, and extend the translations, thereby presenting them in a uniform, consistent, and complete way.

- My theses (Pavlović, 2019, 2020) did not theoretically investigate the correctness of the translation. Thus, Section 6.4 studies the semantics of the considered SPARQL features, based on which it extensively proves the correctness of our translation.

- The prototypical SparqLog system of (Pavlović, 2020) was not optimized and missed some vital SPARQL features. This chapter tackles both problems, allowing SparqLog to be compliant with the standard on all benchmarks (see Section 6.6.2).

- Measuring the query answering performance of SparqLog's prototype was considered out of scope in (Pavlović, 2020). Thus, Section 6.6.3 analyzes this performance and compares SparqLog with the state of the art. The analysis reveals that SparqLog consistently outperforms Fuseki and is highly competitive with Virtuoso while being able to (correctly) answer many more property path queries than both systems.

- Finally, also the ontological reasoning capabilities of SparqLog's prototype were not studied in (Pavlović, 2020). Thus, Section 6.6.4 (*i*) extends SP2Bench with ontological concepts, and (*ii*) subsequently uses it to empirically evaluate SparqLog's and Stardog's reasoning performance. It finds that SparqLog is faster than Stardog on most queries, especially for recursive property path queries with two variables.

6.2 The SparqLog System

This section introduces SparqLog, a system that allows the evaluation of SPARQL 1.1 queries on top of the Vadalog system. To the best of our knowledge, SparqLog is the first system that provides a complete translation engine from SPARQL 1.1 with bag semantics to Datalog. In order to obtain a functional and efficient system, we combined the knowledge provided by the theoretical work with database implementation techniques.

SparqLog implements three translation methods: (i) a *data translation method* T_D which generates Datalog$^{\pm}$ rules from an RDF Dataset; (ii) a *query translation* method T_Q which generates Datalog$^{\pm}$ rules from a SPARQL query; and (iii) a *solution translation method* T_S which generates a SPARQL solution from a Datalog$^{\pm}$ solution. Hence, given an RDF dataset D and a SPARQL query Q, SparqLog generates a Datalog$^{\pm}$ program Π as the union of the rules returned by T_D and T_Q, then evaluates the program Π, and uses T_S to transform the resulting Datalog$^{\pm}$ solution into a SPARQL solution.

6.2.1 Example of Graph Pattern Translation

In order to give a general idea of the translation, we will sketch the translation of the RDF graph and the SPARQL query presented in Section 2.2. To facilitate the notation, we will abbreviate the IRIs using their prefix-based representation. For example, the IRI `http://ex.org/name` will be represented as `ex:name`, where `ex` is a prefix bound to the namespace `http://ex.org/`. Additionally, we will use `film.rdf` instead of `http://ex.org/film.rdf`.

Data translation

Consider the RDF graph G presented in Section 2.2 and restated below for the convenience of the reader:

```
<http://ex.org/glucas> <http://ex.org/name> "George"
<http://ex.org/glucas> <http://ex.org/lastname> "Lucas"
<http://ex.org/rwilliams> <http://ex.org/name> "Robin"
_:b1 <http://ex.org/name> "Robin"
```

First, the data translation method T_D generates a special fact for every RDF term (i.e., IRI, literal, and blank node) in G:

```
iri("ex:glucas"). iri("ex:name"). iri("ex:lastname").
literal("George"). literal("Lucas"). literal("Steven").
bnode("b1").
```

These facts are complemented by the following rules, which represent the domain of RDF terms:

```
term(X) :- iri(X).
term(X) :- literal(X).
term(X) :- bnode(X).
```

For each RDF triple `(s,p,o)` in graph G with IRI `g`, T_D generates a fact `triple(s,p,o,g)`. Hence, in our example, T_D produces:

```
triple("ex:glucas", "ex:name", "George", "film.rdf").
triple("ex:glucas", "ex:lastname", "Lucas", "film.rdf").
triple("ex:rwilliams", "ex:name", "Robin", "film.rdf").
triple("b1", "ex:name", "Robin", "film.rdf").
```

Query translation

Assume that Q is the SPARQL query of Section 2.2, restated in Figure 6.1 for the convenience of the reader. Applying the query translation method T_Q over Q returns the Datalog$^\pm$ rules shown in Figure 6.2. The general principles of the translation will be discussed in Section 6.3. In the interest of readability, we slightly simplify the presentation, e.g., by omitting language tags and type definitions and using simple (intuitive) variable names (rather than more complex ones as would be generated by SparqLog to rule out name clashes).

The query translation method T_Q produces rules for each language construct of SPARQL 1.1 plus rules defining several auxiliary predicates. In addition, also system instructions (e.g., to indicate the answer predicate or ordering requirements) are generated. The translation begins with the `WHERE` clause, then continues with the `SELECT` clause, and finalizes with the `ORDER BY` clause.

```
SELECT ?N ?L
FROM <http://ex.org/film.rdf>
WHERE { ?X <http://ex.org/name> ?N
          . OPTIONAL { ?X <http://ex.org/lastname> ?L }}
ORDER BY ?N
```

Figure 6.1: Example of a SPARQL query.

```
1   // SELECT ?N ?L
2   ans(ID, L, N, D) :- ans1(ID1, L, N, X, D),
3     ID = ["f", L, N, X, ID1].
4   // P1 = { P2 . OPTIONAL { P3 } }
5   ans1(ID1, V2_L, N, X, D) :- ans2(ID2, N, X, D),
6      ans3(ID3, V2_L, V2_X, D), comp(X, V2_X, X),
7      ID1 = ["f1a", X, N, V2_X, V2_L, ID2, ID3].
8   ans1(ID1, L, N, X, D) :- ans2(ID2, N, X, D),
9      not ans_opt1(N, X, D), null(L),
10     ID1 = ["f1b", L, N, X, ID2].
11  ans_opt1(N, X, D) :- ans2(ID2, N, X, D),
12     ans3(ID3, V2_L, V2_X, D), comp(X, V2_X, X).
13  // P2 = ?X ex:name ?N
14  ans2(ID2, N, X, D) :-
15     triple(X, "ex:name",  N, D),
16     D = "default",
17     ID2 = ["f2", X, "ex:name",  N, D].
18  // P3 = ?X ex:lastname ?L
19  ans3(ID3, L, X, D) :-
20     triple(X, "ex:lastname",  L, D),
21     D = "default",
22     ID3 = ["f3", X, "ex:lastname",  L, D].
23  @post("ans", "orderby(2)").
24  @output("ans").
```

Figure 6.2: Datalog$^{\pm}$ rules for the SPARQL query Q of Figure 6.1.

The most complex part of T_Q is the translation of the graph pattern defined in the `WHERE` clause. In our example, the graph pattern defined by the `WHERE` clause is of the form $P_1 = P_2$ `OPTIONAL` P_3 with triple patterns $P_2 =$ `?X ex:name ?N` and $P_3 =$ `?X ex:lastname ?L`. The instruction `@output` (Line 24) is used to define the literal of the goal predicate `ans`. It realizes the projection defined by the `SELECT` clause. The instruction `@post("ans","orderby(2)")` (Line 23) realizes the `ORDER BY` clause; it indicates a sort operation over the elements in the second position of the goal predicate `ans(ID,L,N,D)`, i.e., sorting by `N` (note that `ID` is at position 0). The `ans` predicate is defined (Lines 2–3) by projecting out the `X` variable from the `ans1` relation, which contains the result of evaluating pattern P_1. The tuple IDs are generated as Skolem terms (Line 3 for `ans`; likewise Lines 7, 10, 17, 22). In this example, we assume that the pattern P_1 and its subpatterns P_2 and P_3 are evaluated over the default graph. This is explicitly defined for the basic graph patterns (Lines 15, 20) and propagated by the last argument `D` of the answer predicates.

The OPTIONAL pattern P_1 gives rise to 3 rules defining the predicate `ans1`: a rule (Lines 11–12) to define the predicate `ans_opt1`, which computes those mappings for pattern P_2 that can be extended to mappings of P_3; a rule (Lines 5–7) to compute those tuples of `ans1` that are obtained by extending mappings of P_2 to mappings of P_3; and finally a rule (Lines 8–10) to compute those tuples of `ans1` that are obtained from mappings of P_2 that have no extension to mappings of P_3. In the latter case, the additional variables of P_3 (here: only variable `L`) are set to `null` (Line 9). The two basic graph patterns P_2 and P_3 are translated to rules for the predicates `ans2` (Lines 14–17) and `ans3` (Lines 19–22) in the obvious way.

Solution translation

The evaluation of the program Π produced by the data translation and query translation methods yields a set of ground atoms for the goal predicate `ans`. In our example, we thus get two ground atoms: `ans(id1, "George","Lucas", "film.rdf")` and `ans(id2, "Steven","null","film.rdf")`. Note that the ground atoms are guaranteed to have pairwise distinct tuple IDs. These ground atoms can be easily translated to the *multiset* of solution mappings by projecting out the tuple ID. Due to the simplicity of our example, we only get a *set* $\{\mu_1, \mu_2\}$ of solution mappings with μ_1(`?N`) = `"George"`, μ_1(`?L`) = `"Lucas"`, and μ_2(`?N`) = `"Steven"`.

6.2.2 Example of Property Path Translation

A property path is a feature of the SPARQL query language that allows the user to query for complex paths between nodes instead of being limited to graph patterns with a fixed structure. SPARQL defines different types of property paths, named: PredicatePath, InversePath, SequencePath, AlternativePath, ZeroOrMorePath, OneOrMorePath, ZeroOrOnePath, and NegatedPropertySet. Next, we present an example of the translation of property paths.

Assume that `<http://ex.org/countries.rdf>` identifies an RDF graph with the following prefixed RDF triples:

```
@prefix ex: <http://ex.org/> .
ex:spain ex:borders ex:france .
ex:france ex:borders ex:belgium .
ex:france ex:borders ex:germany .
ex:belgium ex:borders ex:germany .
ex:germany ex:borders ex:austria
```

Note that each triple describes two bordered countries in Europe. Recall that `ex` is a prefix for the namespace `http://ex.org/`, meaning, e.g., that `ex:spain` is the abbreviation of `http://ex.org/spain`.

A natural query could be asking for the countries that can be visited by starting a trip in *Spain*. In other words, we want to get the nodes (countries) reachable from the node representing *Spain*. Although the above query could be expressed by computing the union of different fixed patterns (i.e., one-country trip, two-country trip, etc.), the appropriate way is to use the SPARQL query shown in Figure 6.3. The result of this query is the set $\{\mu_1, \mu_2, \mu_3, \mu_4\}$ of mappings with $\mu_1(?B)$ = `ex:france`, $\mu_2(?B)$ = `ex:germany`, $\mu_3(?B)$ = `ex:austria`, and $\mu_4(?B)$ = `ex:belgium`.

```
PREFIX ex: <http://ex.org/>
SELECT ?B
FROM <http://ex.org/countries.rdf>
WHERE { ?A ex:borders+ ?B . FILTER (?A = ex:spain) }
```

Figure 6.3: Example of a SPARQL property path query.

A property path pattern is a generalization of a triple pattern (s, p, o) where the predicate p is extended to be a regular expression called a property path expression. Hence, the expression `?A ex:borders+ ?B` shown in Figure 6.3 is a property path pattern, where the property path expression `ex:borders+` allows to return all the nodes `?B` reachable from node `?A` by following one or more matches of edges with the `ex:borders` label. The `FILTER` condition restricts the solution mappings to those where variable $?A$ is bound to `ex:spain`, i.e., pairs of nodes where the source node is *spain*. Finally, the `SELECT` clause projects the result to variable `?B`, i.e., the target nodes.

```
1  // P1 = "{?A ex:borders+ ?B . FILTER (?A = ex:spain)}"
2  ans1(ID1,A,B,D) :- ans2(ID2,A,B,D),
3                     X = "ex:spain", ID1 = [...].
4  // P2 = "?A ex:borders+ ?B"
5  ans2(ID2,X,Y,D) :- ans3(ID3,X,Y,D), ID2 = [...].
6  // PP3 = "ex:borders+"
7  ans3(ID3,X,Y,D) :- ans4(ID4,X,Y,D), ID4 = [ ].
8  ans3(ID3,X,Z,D) :- ans4(ID4,X,Y,D),
9                     ans3(ID31,Y,Z,D), ID4 = [ ].
10 // PP4 = "ex:borders"
11 ans4(ID4,X,Y,D) :- triple(X,"ex:borders",Y,D),
12                    D = "default", ID4 = [...].
13 @output("ans1").
```

Figure 6.4: Datalog$^{\pm}$ rules obtained after translating the SPARQL property path query shown in Figure 6.3.

In Figure 6.4, we show the Datalog$^{\pm}$ rules obtained by translating the graph pattern shown in Figure 6.3. The rule in Line 2 corresponds to the translation of the filter graph pattern. The rule in Line 5 is the translation of the property path pattern `?A ex:borders+ ?B`. The rules shown in Lines 8 and 9 demonstrate the use of recursion to emulate the property path expression `ex:borders+`. The rule in Line 11 is the translation of `ex:borders`, called a link property path expression. The general principles of the translation of property paths will be discussed in Section 6.3.3.

6.2.3 Coverage of SPARQL 1.1 Features

In order to develop a realistic integration framework between SPARQL and Vadalog, we conduct a prioritization of SPARQL features. We first lay our focus on basic features, such as *terms* and *graph patterns*. Next, we prepare a more detailed prioritization by considering the results of Bonifati et al. (2020), who examined the real-world adoption of SPARQL features by analyzing a massive amount of real-world query logs from different well-established Semantic Web sources. Additionally, we study further interesting properties of SPARQL, for instance, SPARQL's approach to support partial recursion (through the addition of property paths) or interesting edge cases (such as the combination of *Filter* and *Optional* features) for which a "special" treatment is required.

The outcome of our prioritization step is shown in Table 6.1. For each feature, we present its real-world usage according to (Bonifati et al., 2020) and its current implementation status in our SparqLog system. The table represents the real-world usage by a percentage value (drawn from (Bonifati et al., 2020)) in the feature usage field, if (Bonifati et al., 2020) covers the feature, "Unknown" if (Bonifati et al., 2020) does not cover it, and "Basic Feature" if we consider the feature as fundamental to SPARQL. Note that some features are supported by SparqLog with minor restrictions, such as ORDER BY, for which we did not re-implement the sorting strategy defined by the SPARQL standard but directly use the sorting strategy employed by the Vadalog system. Table 6.1 reveals that our SparqLog engine covers all features that are used in more than 5% of the queries in practice and are deemed, therefore, to be of the highest relevance to SPARQL users. Some of these features have relatively low usage in practice ($< 1\%$); however, they are still supported by our engine. These features include *property paths* and GROUP BY. We have chosen to add *property paths* to our engine, as they are not only interesting for being SPARQL's approach to support partial recursion but, according to (Bonifati et al., 2020), some datasets make extensive use of them. Moreover, we have chosen to add GROUP BY and some aggregates (e.g., COUNT), as they are critical in traditional database settings and, thus, are essential to establish a bridge between the Semantic Web and Database communities.

In addition to these most widely used features, SparqLog covers all features occurring in critical benchmarks (see Section 6.1 for a detailed discussion). Specifically, as used in the FEASIBLE benchmark, SparqLog covers the following features: ORDER BY with complex arguments (such as ORDER BY with BOUND conditions), functions on strings such as UCASE, the DATATYPE function, LIMIT, and OFFSET. For the gMark benchmark, we cover the "exactly n occurrences" property path, the "n or more occurrences" property path, and the "between 0 and n occurrences" property path.

General Feature	Specific Feature	Feature Usage	Status
Terms	IRIs, Literals, Blank nodes	Basic Feature	✓
Semantics	Sets, Bags	Basic Feature	✓
Graph patterns	Triple pattern	Basic Feature	✓
	AND / JOIN	28.25%	✓
	OPTIONAL	16.21%	✓
	UNION	18.63%	✓
	GROUP Graph Pattern	< 1%	✗
Filter constraints	Equality / Inequality	All Constraints 40.15%	✓
	Arithmetic Comparison		✓
	bound, isIRI, isBlank, isLiteral		✓
	Regex		✓
	AND, OR, NOT		✓
Query forms	SELECT	87.97%	✓
	ASK	4.97%	✓
	CONSTRUCT	4.49%	✗
	DESCRIBE	2.47%	✗
Solution modifiers	ORDER BY	2.06%	✓
	DISTINCT	21.72%	✓
	LIMIT	17.00%	✓
	OFFSET	6.15%	✓
RDF datasets	GRAPH ?x { ... }	2.71%	✓
	FROM (NAMED)	Unknown	✗
Negation	MINUS	1.36%	✓
	FILTER NOT EXISTS	1.65%	✗
Property paths	LinkPath (X exp Y)	< 1%	✓
	InversePath (^exp)	< 1%	✓
	SequencePath (exp1 / exp2)	< 1%	✓
	AlternativePath (exp1 | exp2)	< 1%	✓
	ZeroOrMorePath (exp*)	< 1%	✓
	OneOrMorePath (exp+)	< 1%	✓
	ZeroOrOnePath (expr?)	< 1%	✓
	NegatedPropertySet (!expr)	< 1%	✓
Assignment	BIND	< 1%	✗
	VALUES	< 1%	✗
Aggregates	GROUP BY	< 1%	✓
	HAVING	< 1%	✗
Sub-Queries	Sub-Select Graph Pattern	< 1%	✗
	FILTER EXISTS	< 1%	✗
Filter functions	Coalesce	Unknown	✗
	IN / NOT IN	Unknown	✗

Table 6.1: Selected SPARQL features, including their real-world usage according to (Bonifati et al., 2020) and the current status in SparqLog.

Among our contributions concerning the translation of SPARQL to Datalog are: the available translation methods have been combined into a uniform and practical framework for translating RDF datasets and SPARQL queries to Warded Datalog± programs; we have developed simpler translations for MINUS and OPT, compared with (Polleres and Wallner, 2013); we provide translations for both bag and set semantics, thus covering queries with and without the DISTINCT keyword; we have enhanced current translations by adding partial support for data types and language tags; we have developed a novel duplicate preservation model based on the abstract theories of ID generation (this was required because plain existential ID generation turned out to be problematic due to its dependence on a very specific chase algorithm of the Vadalog system), and we propose a complete method for translating property paths, including zero-or-one and zero-or-more property paths.

There are also a few features that have a real-world usage of slightly above one percent and which are currently not supported by SparqLog. Among these features are CONSTRUCT, DESCRIBE, and FILTER NOT EXISTS. We do not support features CONSTRUCT and DESCRIBE, as these solution modifiers do not yield any interesting theoretical or practical challenges, and they did not occur in any of the benchmarks chosen for our experimental evaluation. The features for query federation are out of the considered scope, as our translation engine demands RDF datasets to be translated to the Vadalog system for query answering. Furthermore, SPARQL query federation is used in less than 1% of SPARQL queries (Bonifati et al., 2020).

6.3 Translating SPARQL 1.1 to Warded Datalog$^{\pm}$

In this section, we provide a detailed description of our translation from SPARQL 1.1 to Warded Datalog$^{\pm}$. Note that many of the rules thus generated are simple Datalog rules, i.e., they do not have existentially quantified variables in the head. In such cases, we shall interchangeably refer to these rules as "'Datalog rules" or "Datalog$^{\pm}$ rules". Of course, if existentially quantified variables are indeed used in the head, we shall always speak of "Datalog$^{\pm}$ rules".

We start by translating RDF graphs to Datalog rules in Section 6.3.1. We then detail our translation of graph patterns and the specific translation rules for property path expressions in Sections 6.3.2 and 6.3.3. Finally, in Section 6.3.4, we consider query forms.

6.3.1 Translation of RDF Graphs

Assume that I, L, and B are disjoint infinite sets corresponding to IRIs, literals, and blank nodes. An RDF term is an element in the set $T = I \cup L \cup B$. An *RDF triple* is a tuple $(s, p, o) \in T \times I \times T$ where s is called the *subject*, p is the *predicate*, and o is the *object*. An *RDF graph* G is a set of RDF triples. An *RDF dataset* D is a collection of graphs including a default graph G_0 and zero or more named graphs, such that a named graph is a pair (u, G) where u is an IRI which identifies the RDF graph G.

Let G be a given RDF graph; the translation of the graph to Datalog facts is defined as follows:

1. For each IRI, constant and blank node in G, the corresponding facts $iri(X)$, $literal(X)$, and $bnode(X)$ are generated.
2. For each named graph g a tuple $named(g)$ and for each triple (s, p, o) of graph g a fact $triple(s, p, o, g)$ where g is either "default" for the default graph, or the IRI of a named graph is created.
3. A term is either an IRI, a literal, or a blank node: The set of terms is represented by the predicate *term*.

Definition 6.3.1 (Terms). *The predicate terms is defined as follows:*

$$term(X) :- \ iri(X).$$
$$term(X) :- \ literal(X).$$
$$term(X) :- \ bnode(X).$$

6.3.2 Translation of SPARQL Graph Patterns

Graph Pattern. Assume the existence of an infinite set V of variables disjoint from T. We will use var(α) to denote the set of variables occurring in any structure α. A *graph pattern* is defined recursively as follows: a tuple from $(T \cup V) \times (I \cup V) \times (T \cup V)$ is a triple pattern; if P_1 and P_2 are graph patterns, C is a filter constraint, and $g \in I$ then $\{P_1 . P_2\}$, $\{P_1 \text{ UNION } P_2\}$, $\{P_1 \text{ OPT } P_2\}$, $\{P_1 \text{ MINUS } P_2\}$, $\{P_1 \text{ FILTER } C\}$, and; $\{\text{GRAPH } g \ P_1\}$ are graph patterns. A *filter constraint* is defined recursively as follows: (i) If $?X, ?Y \in V$, $c \in I \cup L$ and r is a regular expression then *true*, *false*, $?X = c$, $?X =?Y$, bound$(?X)$, isIRI$(?X)$, isBlank$(?X)$, isLiteral$(?X)$ and regex$(?X, r)$ are *atomic filter constraints*; (ii) If C_1 and C_2 are filter constraints then $(!C_1)$, $(C_1 \ \&\& \ C_2)$ and $(C_1 \ || \ C_2)$ are *Boolean filter constraints.*

Subpattern. A *subpattern* P' of a graph pattern P is defined to be any substring of P that is also a graph pattern. Furthermore, P' is defined to be an immediate subpattern of P if it is a subpattern of P and if there is no other subpattern of P, different from P, that contains P'. A *parse tree* is specified as a tree $< V, E >$ with the set of nodes V being the subpatterns of a graph pattern P and the set of edges E containing an edge (P_1, P_2) if P_2 is an immediate subpattern of P_1.

Solution Mapping. The evaluation of a graph pattern results in a multiset of solution mappings. A *solution mapping* is a partial function $\mu : V \to T$, i.e., an assignment of variables to RDF terms. The domain of μ, denoted dom(μ), is the subset of V where μ is defined. The *empty mapping*, denoted μ_0, is the mapping satisfying that dom$(\mu_0) = \emptyset$. A *multiset of solution mappings* Ω is an unsorted list of solution mappings where duplicates are allowed. The domain of Ω is the set of variables occurring in the solution mappings of Ω.

Translation Function. Let P be a SPARQL graph pattern and D be an RDF dataset $D = \langle G, G_{named} \rangle$ where G is the default graph and G_{named} is the set of named graphs. The translation of graph patterns is realized by the translation function $\tau(P, dst, D, NodeIndex)$ where: P is the graph pattern that should be translated next; dst (short for "distinct") is a Boolean value that describes whether the result should have set semantics ($dst = true$) or bag semantics ($dst = false$); D is the graph on which the pattern should be evaluated; $NodeIndex$ is the index of the pattern P to be translated; and the output of function τ is a set of Datalog$^{\pm}$ rules. In the following paragraphs, we briefly discuss some aspects of the translation before stating τ's definitions.

General Strategy of the Translation. Analogously to (Polleres, 2007; Polleres and Wallner, 2013), our translation proceeds by recursively traversing the parse tree of a SPARQL 1.1 query and translating each subpattern into its respective Datalog$^{\pm}$ rules. Subpatterns of the parse tree are indexed. The root has index 1, the left child of the i-th node has index $2 * i$, and the right child has index $2 * i + 1$. During the translation, bindings of the i-th subpattern are represented by the predicate ans_i. In all answer predicates ans_i, we have the current graph as the last component. It can be changed by the GRAPH construct; for all other SPARQL constructs, it is transparently passed on from the children to the parent in the parse tree. Since the order of variables in predicates is relevant, some variable sets will need to be lexicographically ordered, which we denote by $\overline{x}$ as in (Polleres and Wallner, 2013). We write $\overline{var}(P)$ to denote the lexicographically ordered tuple of variables of P. Moreover, a variable renaming function $v_j : V \rightarrow V$ is defined.

Auxiliary Predicates. The translation generates several auxiliary predicates. Above all, we need a predicate *comp* for testing if two mappings are *compatible*. The notion of compatible mappings is fundamental to the evaluation of SPARQL graph patterns. Two mappings μ_1 and μ_2 are *compatible*, denoted $\mu_1 \sim \mu_2$, if for all $?X \in \mathrm{dom}(\mu_1) \cap \mathrm{dom}(\mu_2)$ it is satisfied that $\mu_1(?X) = \mu_2(?X)$. The auxiliary predicate $comp(X_1, X_2, X_3)$ checks if two values X_1 and X_2 are compatible. The third position X_3 represents the value that is used in the result tuple when joining over X_1 and X_2:

$$\begin{aligned}
&null(\text{"}null\text{"}).\\
comp(X, X, X) &:- \; term(X).\\
comp(X, Z, X) &:- \; term(X), null(Z).\\
comp(Z, X, X) &:- \; term(X), null(Z).\\
comp(Z, Z, Z) &:- \; null(Z).
\end{aligned}$$

Bag Semantics. For bag semantics (i.e., $dst = false$), all answer predicates contain a fresh existential variable when they occur in the head of a rule. In this way, whenever such a rule fires, a fresh tuple ID is generated. This is particularly important for the translation of the UNION construct. In contrast to (Polleres and Wallner, 2013), we can thus distinguish duplicates without the need to increase the arity of the answer predicate. We have developed a novel duplicate preservation model based on the abstract theories of ID generation of (Bertossi et al., 2019). As mentioned above, plain existential ID generation turned out to be problematic due to the peculiarities of the Vadalog system. Therefore, our ID generation process is abstracted away using a Skolem function generator and representing nulls (corresponding to tuple IDs) as specific Skolem terms.

Filter Constraints. Note how we treat filter conditions in FILTER constructs: building our translation engine on top of the Vadalog system allows us to literally copy (possibly complex) filter conditions into the rule body and let the Vadalog system evaluate them. For instance, the regex functionality uses the corresponding Vadalog function, which makes direct use of the Java regex library. For evaluating filter functions, such as isIRI, isURI, isBlank, isLiteral, isNumeric, and bound expressions, our translation engine uses the corresponding auxiliary predicates generated in our data translation method.

This finishes the discussion of τ. In the following, we list the translation of various graph patterns in Definitions 6.3.2 and 6.3.10. To improve readability, we omit the explicit generation of IDs via Skolem functions and put a fresh ID-variable in the first position of the head atoms of the rules.

Definition 6.3.2 (Triple)**.** *Let P_i be the i-th subpattern of P and furthermore let P_i be a triple pattern (s, p, o), then $\tau(P_i, true, D, i)$ is defined as:*

$$ans_i(\overline{var}(P_i), D) :- \ triple(s, p, o, D).$$

And $\tau(P_i, false, D, i)$ is defined as:

$$ans_i(Id, \overline{var}(P_i), D) :- \ triple(s, p, o, D).$$

Definition 6.3.3 (Graph)**.** *Let P_i be the i-th subpattern of P and furthermore let P_i be* (GRAPH g P_1)*, then $\tau(P_i, true, D, i)$ is defined as:*

$$\begin{aligned} ans_i(\overline{var}(P_i), D) :- \ & ans_{2i}(\overline{var}(P_1), g), \\ & named(g). \\ \tau(P_1, true, g, 2i). & \end{aligned}$$

And $\tau(P_i, false, D, i)$ is defined as:

$$\begin{aligned} ans_i(Id, \overline{var}(P_i), D) :- \ & ans_{2i}(Id_1, \overline{var}(P_1), g), \\ & named(g). \\ \tau(P_1, false, g, 2i) & \end{aligned}$$

Definition 6.3.4 (Join)**.** *Let P_i be the i-th subpattern of P and furthermore let P_i be $(P_1 \,.\, P_2)$, then $\tau(P_i, true, D, i)$ is defined as:*

$$\begin{aligned} ans_i(\overline{var}(P_i), D) :- \; & ans_{2i}(v_1(\overline{var}(P_1)), D),\\ & ans_{2i+1}(v_2(\overline{var}(P_2)), D),\\ & comp(v_1(x_1), v_2(x_1), x_1), \ldots, comp(v_1(x_n), v_2(x_n), x_n).\end{aligned}$$

$$\begin{aligned} \tau(P_1, true, D, 2i)\\ \tau(P_2, true, D, 2i+1)\end{aligned}$$

And $\tau(P_i, false, D, i)$ is defined as:

$$\begin{aligned} ans_i(Id, \overline{var}(P_i), D) :- \; & ans_{2i}(Id_1, v_1(\overline{var}(P_1)), D),\\ & ans_{2i+1}(Id_2, v_2(\overline{var}(P_2)), D),\\ & comp(v_1(x_1), v_2(x_1), x_1), \ldots, comp(v_1(x_n), v_2(x_n), x_n).\end{aligned}$$

$$\begin{aligned} \tau(P_1, false, D, 2i)\\ \tau(P_2, false, D, 2i+1)\end{aligned}$$

Here, we are using the following notation:

- $\overline{var}(P_i) = \overline{var(P_1) \cup var(P_2)}$
- $\{x_1, \ldots, x_n\} = var(P_1) \cap var(P_2)$
- $v_1, v_2 : var(P_1) \cap var(P_2) \to V$, *such that* $Image(v_1) \cap Image(v_2) = \emptyset$

Definition 6.3.5 (Union)**.** *Let P_i be the i-th subpattern of P and furthermore let P_i be $(P_1 \text{ UNION } P_2)$, then $\tau(P_i, true, D, i)$ is defined as:*

$$\begin{aligned} ans_i(\overline{var}(P_i), D) :- \; & ans_{2i}(\overline{var}(P_1), D),\\ & null(x_1), \ldots null(x_n).\\ ans_i(\overline{var}(P_i), D) :- \; & ans_{2i+1}(\overline{var}(P_2), D),\\ & null(y_1), \ldots null(y_m).\end{aligned}$$

$$\begin{aligned} \tau(P_1, true, D, 2i)\\ \tau(P_2, true, D, 2i+1)\end{aligned}$$

And $\tau(P_i, false, D, i)$ is defined as:

$$
\begin{aligned}
ans_i(Id, \overline{var}(P_i), D) :- \; & ans_{2i}(Id_1, \overline{var}(P_1), D),\\
& null(x_1), \ldots null(x_n).\\
ans_i(Id, \overline{var}(P_i), D) :- \; & ans_{2i+1}(Id_2, \overline{var}(P_2), D),\\
& null(y_1), \ldots null(y_m).
\end{aligned}
$$

$$
\begin{aligned}
\tau(P_1, false, D, 2i)\\
\tau(P_2, false, D, 2i+1)
\end{aligned}
$$

Here, we are using the following notation:

- $\{x_1, \ldots, x_n\} = var(P_2) \setminus var(P_1)$
- $\{y_1, \ldots, y_m\} = var(P_1) \setminus var(P_2)$

Definition 6.3.6 (Optional). *Let P_i be the i-th subpattern of P and furthermore let P_i be $(P_1 \, \mathrm{OPT} \, P_2)$, then $\tau(P_i, true, D, i)$ is defined as:*

$$
\begin{aligned}
ans_{opt-i}(\overline{var}(P_1), D) :- \; & ans_{2i}(\overline{var}(P_1), D),\\
& ans_{2i+1}(v_2(\overline{var}(P_2)), D),\\
& comp(x_1, v_2(x_1), z_1), \ldots, comp(x_n, v_2(x_n), z_n).
\end{aligned}
$$

$$
\begin{aligned}
ans_i(\overline{var}(P_i), D) :- \; & ans_{2i}(v_1(\overline{var}(P_1)), D),\\
& ans_{2i+1}(v_2(\overline{var}(P_2)), D),\\
& comp(v_1(x_1), v_2(x_1), x_1), \ldots, comp(v_1(x_n), v_2(x_n), x_n).
\end{aligned}
$$

$$
\begin{aligned}
ans_i(\overline{var}(P_i), D) :- \; & ans_{2i}(\overline{var}(P_1), D),\\
& not \; ans_{opt-i}(\overline{var}(P_1), D),\\
& null(y_1), \ldots, null(y_m).
\end{aligned}
$$

$$
\begin{aligned}
\tau(P_1, true, D, 2i)\\
\tau(P_2, true, D, 2i+1)
\end{aligned}
$$

And $\tau(P_i, \mathit{false}, D, i)$ *is defined as:*

$$\begin{aligned} ans_{opt-i}(\overline{var}(P_1), D) :- \ & ans_{2i}(Id_1, \overline{var}(P_1), D), \\ & ans_{2i+1}(Id_2, v_2(\overline{var}(P_2)), D), \\ & comp(x_1, v_2(x_1), z_1), \dots, comp(x_n, v_2(x_n), z_n). \end{aligned}$$

$$\begin{aligned} ans_i(Id, \overline{var}(P_i), D) :- \ & ans_{2i}(Id_1, v_1(\overline{var}(P_1)), D), \\ & ans_{2i+1}(Id_2, v_2(\overline{var}(P_2)), D), \\ & comp(v_1(x_1), v_2(x_1), x_1), \dots, comp(v_1(x_n), v_2(x_n), x_n). \end{aligned}$$

$$\begin{aligned} ans_i(Id, \overline{var}(P_i), D) :- \ & ans_{2i}(Id_1, \overline{var}(P_1), D), \\ & not\ ans_{opt-i}(\overline{var}(P_1), D), \\ & null(y_1), \dots, null(y_m). \end{aligned}$$

$$\begin{aligned} &\tau(P_1, \mathit{false}, D, 2i) \\ &\tau(P_2, \mathit{false}, D, 2i+1) \end{aligned}$$

Here, we are using the following notation:

- $\overline{var}(P_i) = \overline{var(P_1) \cup var(P_2)}$
- $\{x_1, \dots, x_n\} = var(P_1) \cap var(P_2)$
- $\{y_1, \dots, y_m\} = var(P_2) \setminus var(P_1)$
- $v_1, v_2 : var(P_1) \cap var(P_2) \rightarrow V$, *such that* $Image(v_1) \cap Image(v_2) = \emptyset$

Definition 6.3.7 (Filter)**.** *Let* P_i *be the i-th subpattern of* P *and furthermore let* P_i *be* $(P_1\, \text{FILTER}\, C)$*, then* $\tau(P_i, \mathit{true}, D, i)$ *is defined as:*

$$\begin{aligned} ans_i(\overline{var}(P_i), D) :- \ & ans_{2i}(\overline{var}(P_1), D), C. \\ &\tau(P_1, \mathit{true}, D, 2i) \end{aligned}$$

And $\tau(P_i, \mathit{false}, D, i)$ *is defined as:*

$$\begin{aligned} ans_i(Id, \overline{var}(P_i), D) :- \ & ans_{2i}(Id_1, \overline{var}(P_1), D), C. \\ &\tau(P_1, \mathit{false}, D, 2i) \end{aligned}$$

Definition 6.3.8 (Optional Filter)**.** *Let P_i be the i-th subpattern of P and furthermore let P_i be $(P_1\,\mathrm{OPT}(P_2\,\mathrm{FILTER}\,C))$, then $\tau(P_i, true, D, i)$ is defined as:*

$$\begin{aligned}
ans_{opt-i}(\overline{var}(P_1), D) :-\ & ans_{2i}(\overline{var}(P_1), D),\\
& ans_{2i+1}(v_2(\overline{var}(P_2)), D),\\
& comp(x_1, v_2(x_1), z_1), \ldots, comp(x_n, v_2(x_n), z_n), C.
\end{aligned}$$

$$\begin{aligned}
ans_i(\overline{var}(P_i), D) :-\ & ans_{2i}(v_1(\overline{var}(P_1)), D),\\
& ans_{2i+1}(v_2(\overline{var}(P_2)), D),\\
& comp(v_1(x_1), v_2(x_1), x_1), \ldots, comp(v_1(x_n), v_2(x_n), x_n), C.
\end{aligned}$$

$$\begin{aligned}
ans_i(\overline{var}(P_i), D) :-\ & ans_{2i}(\overline{var}(P_1), D),\\
& not\ ans_{opt-i}(\overline{var}(P_1), D),\\
& null(y_1), \ldots, null(y_m).
\end{aligned}$$

$$\begin{aligned}
\tau(P_1, true, D, 2i)\\
\tau(P_2, true, D, 2i+1)
\end{aligned}$$

And $\tau(P_i, false, D, i)$ is defined as:

$$\begin{aligned}
ans_{opt-i}(\overline{var}(P_1), D) :-\ & ans_{2i}(Id_1, \overline{var}(P_1), D),\\
& ans_{2i+1}(Id_2, v_2(\overline{var}(P_2)), D),\\
& comp(x_1, v_2(x_1), z_1), \ldots, comp(x_n, v_2(x_n), z_n), C.
\end{aligned}$$

$$\begin{aligned}
ans_i(Id, \overline{var}(P_i), D) :-\ & ans_{2i}(Id_1, v_1(\overline{var}(P_1)), D),\\
& ans_{2i+1}(Id_2, v_2(\overline{var}(P_2)), D),\\
& comp(v_1(x_1), v_2(x_1), x_1), \ldots, comp(v_1(x_n), v_2(x_n), x_n), C.
\end{aligned}$$

$$\begin{aligned}
ans_i(Id, \overline{var}(P_i), D) :-\ & ans_{2i}(Id_1, \overline{var}(P_1), D),\\
& not\ ans_{opt-i}(\overline{var}(P_1), D),\\
& null(y_1), \ldots, null(y_m).
\end{aligned}$$

$$\begin{aligned}
\tau(P_1, false, D, 2i)\\
\tau(P_2, false, D, 2i+1)
\end{aligned}$$

Here, we are using the following notation:

- $\overline{var}(P_i) = \overline{var(P_1) \cup var(P_2)}$
- $\{x_1, \ldots, x_n\} = var(P_1) \cap var(P_2)$
- $\{y_1, \ldots, y_m\} = var(P_2) \setminus var(P_1)$
- $v_1, v_2 : var(P_1) \cap var(P_2) \to V$*, such that* $Image(v_1) \cap Image(v_2) = \emptyset$

Definition 6.3.9 (Minus)**.** *Let P_i be the i-th subpattern of P and furthermore let P_i be* $(P_1 \,\text{MINUS}\, P_2)$*, then $\tau(P_i, true, D, i)$ is defined as:*

$$\begin{aligned}
ans_{join-i}(\overline{var}(P_i), D) :- \;& ans_{2i}(\overline{var}(P_1), D),\\
& ans_{2i+1}(v_2(\overline{var}(P_2)), D),\\
& comp(x_1, v_2(x_1), z_1), \ldots, comp(x_n, v_2(x_n), z_n).\\
ans_{equal-i}(\overline{var}(P_1), D) :- \;& ans_{join-i}(\overline{var}(P_i), D),\\
& x_1 = v_2(x_1),\; not\ null(x_1).\\
& \ldots\\
ans_{equal-i}(\overline{var}(P_1), D) :- \;& ans_{join-i}(\overline{var}(P_i), D),\\
& x_n = v_2(x_n),\; not\ null(x_n).\\
ans_i(\overline{var}(P_1), D) :- \;& ans_{2i}(\overline{var}(P_1), D),\\
& not\ ans_{equal-i}(\overline{var}(P_1), D).
\end{aligned}$$

$$\begin{aligned}
\tau(P_1, true, D, 2i)\\
\tau(P_2, true, D, 2i+1)
\end{aligned}$$

And $\tau(P_i, false, D, i)$ is defined as:

$$\begin{aligned}
ans_{join-i}(\overline{var}(P_i), D) :- \;& ans_{2i}(Id_1, \overline{var}(P_1), D),\\
& ans_{2i+1}(Id_2, v_2(\overline{var}(P_2)), D),\\
& comp(x_1, v_2(x_1), z_1), \ldots, comp(x_n, v_2(x_n), z_n).\\
ans_{equal-i}(\overline{var}(P_1), D) :- \;& ans_{join-i}(\overline{var}(P_i), D),\\
& x_1 = v_2(x_1),\; not\ null(x_1).\\
& \ldots\\
ans_{equal-i}(\overline{var}(P_1), D) :- \;& ans_{join-i}(\overline{var}(P_i), D),\\
& x_n = v_2(x_n),\; not\ null(x_n).\\
ans_i(Id, \overline{var}(P_1), D) :- \;& ans_{2i}(Id_1, \overline{var}(P_1), D),\\
& not\ ans_{equal-i}(\overline{var}(P_1), D).
\end{aligned}$$

$$\begin{aligned}
\tau(P_1, false, D, 2i)\\
\tau(P_2, false, D, 2i+1)
\end{aligned}$$

Here, we are using the following notation:

- $\overline{var}(P_i) = \overline{var(P_1) \cup v_2(var(P_2))}$
- $\{x_1, \ldots, x_n\} = var(P_1) \cap var(P_2)$
- $v_2 : var(P_1) \cap var(P_2) \to V \setminus var(P_1)$

Property path patterns are given in the form $S\ P_1\ O$, where P_1 is a property path expression. Due to the complex semantics of property paths, we have introduced a separate translation function τ_{PP} for property path expressions, which we will take a closer look at in Section 6.3.3.

Definition 6.3.10 (Property Path Pattern)**.** *Let P_i be the i-th subpattern of P and let $P_i = S\ P_1\ O$ be a property path pattern, then $\tau(P_i, true, D, i)$ is defined as:*

$$\begin{aligned} ans_i(\overline{var}(P_i), D) &:- \ ans_{2i}(S, O, D). \\ \tau_{PP}(P_1, true, S, O, D, 2i) & \end{aligned}$$

And $\tau(P_i, false, D, i)$ is defined as:

$$\begin{aligned} ans_i(Id, \overline{var}(P_i), D) &:- \ ans_{2i}(Id_1, S, O, D). \\ \tau_{PP}(P_1, false, S, O, D, 2i) & \end{aligned}$$

with τ_{PP} being the translation function for property path expressions, defined next.

6.3.3 Translation of Property Path Expressions

Property paths are an important feature introduced in SPARQL 1.1. A translation of property paths to Datalog was presented in (Polleres and Wallner, 2013) – but not fully compliant with the SPARQL 1.1 standard: the main problem in (Polleres and Wallner, 2013) was the way how *zero-or-one* and *zero-or-more* property paths were handled. In particular, the case that a path of zero length from t to t also exists for those terms t which occur in the query but not in the current graph, was omitted in (Polleres and Wallner, 2013). A *property path pattern* is given in the form s, p, o, where s, o are the usual subject and object, and p is a *property path expression*. That is, p is either an IRI (the base case) or composed from one or two other property path expressions p_1, p_2 as: $\hat{}p_1$ (inverse path expression), $p_1 \mid p_2$ (alternative path expression), p_1/p_2 (sequence path expression), $p_1?$ (zero-or-one path expression), p_1+ (one-or-more path expression), p_1* (zero-or-more path expression), or $!p_1$ (negated path expression). A property path pattern s, p, o is translated by first translating the property path expression p into rules for each subexpression of p. The end points s and o of the overall path are only applied to the top-level expression p. Analogously to our translation function $\tau(P, dst, D, NodeIndex)$ for graph patterns, we now also introduce a translation function $\tau_{PP}(PP, dst, S, O, D, NodeIndex)$ for property path expressions PP, where S, O, are the subject and object of the top-level property path expression that have to be kept track of during the entire evaluation (cf., Definition 6.3.10).

The translation of a property path pattern S, P_1, O for some property path expression P_1 consists of two parts: the translation of P_1 by the translation function τ_{PP} and the translation τ of S, P_1, O (see Definition 6.3.10) – now applying the end points S and O to the top-level property path expression P_1. The base case of τ_{PP} is a link property path $PP_i = p_1$ (i.e., simply an IRI), which returns all pairs (X, Y) that occur as subject and object in a triple with predicate p_1 (see Definition 6.3.11). Equally simple translations apply to inverse paths, which swap start point and end point (see Definition 6.3.12); alternative paths, which are treated similarly to UNION in Definition 6.3.5 (see Definition 6.3.13); and sequence paths, which combine two paths by identifying the end point of the first path with the start point of the second path (see Definition 6.3.14). In the following, we will only state the translations for property path expressions under bag semantics (i.e., $dst = false$), since for set semantics, the IDs are simply left out or set to a constant value (e.g., $Id = []$).

Definition 6.3.11 (Link Property Path)**.** *Let PP_i be the i-th subexpression of a property path expression PP and furthermore let $PP_i = p_1$ be a link property path expression. Then $\tau_{PP}(PP_i, false, S, O, D, i)$ is defined as:*

$$ans_i(Id, X, Y, D) :- \ triple(X, p_1, Y, D).$$

Definition 6.3.12 (Inverse Property Path)**.** *Let PP_i be the i-th subexpression of a property path expression PP and furthermore let $PP_i = \hat{}PP_1$ be an inverse property path expression. Then $\tau_{PP}(PP_i, false, S, O, D, i)$ is defined as:*

$$\begin{aligned} ans_i(Id, X, Y, D) &:- \ ans_{2i}(Id_1, Y, X, D). \\ \tau_{PP}(PP_1, false, S, O, D, 2i) & \end{aligned}$$

Definition 6.3.13 (Alternative Property Path)**.** *Let PP_i be the i-th subexpression of a property path expression PP and furthermore let $PP_i = PP_1|PP_2$ be an alternative property path expression. Then $\tau_{PP}(PP_i, false, S, O, D, i)$ is defined as:*

$$\begin{aligned} ans_i(Id, X, Y, D) &:- \ ans_{2i}(Id_1, X, Y, D). \\ ans_i(Id, X, Y, D) &:- \ ans_{2i+1}(Id_1, X, Y, D). \\ \tau_{PP}(PP_1, false, S, O, D, 2i) & \\ \tau_{PP}(PP_2, false, S, O, D, 2i+1) & \end{aligned}$$

Definition 6.3.14 (Sequence Property Path)**.** *Let PP_i be the i-th subexpression of a property path expression PP and furthermore let $PP_i = PP_1/PP_2$ be a sequence property path expression. Then $\tau_{PP}(PP_i, false, S, O, D, i)$ is defined as:*

$$\begin{aligned} ans_i(Id, X, Z, D) &:- \ ans_{2i}(Id_1, X, Y, D), \\ & \quad\ ans_{2i+1}(Id_2, Y, Z, D). \\ \tau_{PP}(PP_1, false, S, O, D, 2i) & \\ \tau_{PP}(PP_2, false, S, O, D, 2i+1) & \end{aligned}$$

For zero-or-one paths (and likewise for zero-or-more paths), we need to collect all terms that occur as subjects or objects in the current graph by an auxiliary predicate `subjectOrObject` (Definition 6.3.15). As shown in Definition 6.3.16, this auxiliary predicate is needed to produce paths of length zero (i.e., from X to X) for all these terms occurring in the current graph. Moreover, if exactly one of S and O is not a variable, or if both are the same non-variable, then also, for these nodes, we have to produce paths of zero length. It is because of this special treatment of zero-length paths that the subject S and object O from the top-level property path expression have to be propagated through all recursive calls of the translation function τ_{PP}. In addition to the zero-length paths, of course, also paths of length one have to be produced by recursively applying the translation τ_{PP} to PP_1 if PP_i is of the form $PP_i = PP_1?$.

Definition 6.3.15 (SubjectOrObject)**.** *The subjectOrObject predicate defines intuitively the set of all possible subjects and objects occurring in a graph, i.e.:*

$$\begin{aligned} \mathit{subjectOrObject}(X) &:- \mathit{triple}(X,P,Y,D).\\ \mathit{subjectOrObject}(Y) &:- \mathit{triple}(X,P,Y,D). \end{aligned}$$

It should be noted that, according to the SPARQL semantics of property paths[1], zero-or-one, zero-or-more, and one-or-more property paths always have set semantics. This is why the Datalog$^\pm$ rules for these three path expressions contain a body literal $\mathit{Id} = []$. By forcing the tuple ID to the same value whenever one of these rules fires, multiply derived tuples are indistinguishable for our system and will, therefore, never give rise to duplicates.

Definition 6.3.16 (Zero-Or-One Property Path)**.** *Let PP_i be the i-th subexpression of a property path expression PP and furthermore let $PP_i = PP_1?$ be a zero-or-one property property path expression. Then $\tau_{PP}(PP_i, \mathit{false}, S, O, D, i)$ consists of the following rules:*

$$\begin{aligned} \mathit{ans}_i(\mathit{Id},X,X,D) &:- \mathit{subjectOrObject}(X), \mathit{Id} = [].\\ \mathit{ans}_i(\mathit{Id},X,Y,D) &:- \mathit{ans}_{2i}(\mathit{Id}_1,X,Y,D), \mathit{Id} = [].\\ \tau_{PP}(PP_1, \mathit{false}, S, O, D, 2i) & \end{aligned}$$

Moreover, if either one of S and O is a variable and the other is a non-variable t or both S and O are the same non-variable t, then the following rule is added:

$$\mathit{ans}_i(\mathit{Id},X,X,D) :- \mathit{not}\ \mathit{term}(X), X = t, \mathit{Id} = [].$$

Furthermore, one-or-more paths are realized in the usual style of transitive closure programs in Datalog (see Definition 6.3.17).

Definition 6.3.17 (One-Or-More Property Path)**.** *Let PP_i be the i-th subexpression of a property path expression PP and furthermore let $PP_i = PP_1+$ be a one-or-more property path expression. Then $\tau_{PP}(PP_i, \mathit{false}, S, O, D, i)$ is defined as:*

$$\begin{aligned} \mathit{ans}_i(\mathit{Id},X,Y,D) &:- \mathit{ans}_{2i}(\mathit{Id}_1,X,Y,D), \mathit{Id} = [].\\ \mathit{ans}_i(\mathit{Id},X,Z,D) &:- \mathit{ans}_{2i}(\mathit{Id}_1,X,Y,D),\\ &\qquad \mathit{ans}_i(\mathit{Id}_2,Y,Z,D), \mathit{Id} = [].\\ \tau_{PP}(PP_1, \mathit{false}, S, O, D, 2i) & \end{aligned}$$

[1] https://www.w3.org/TR/SPARQL11-query/#defn_PropertyPathExpr (last visited 09/25/2023)

Essentially, the zero-or-more property path (see Definition 6.3.18) is a combination of the zero-or-one and one-or-more property paths.

Definition 6.3.18 (Zero-Or-More Property Path)**.** *Let PP_i be the i-th subexpression of a property path expression PP and furthermore let $PP_i = PP_1*$ be a zero-or-more property property path expression. Then $\tau_{PP}(PP_i, false, S, O, D, i)$ consists of the following rules:*

$$\begin{aligned}
ans_i(Id, X, X, D) &:- \; subjectOrObject(X), Id = [].\\
ans_i(Id, X, Y, D) &:- \; ans_{2i}(Id_1, X, Y, D), Id = [].\\
ans_i(Id, X, Z, D) &:- \; ans_{2i}(Id_1, X, Y, D),\\
&\qquad ans_i(Id_2, Y, Z, D), Id = [].\\
\tau_{PP}(PP_1, false, S, O, D, 2i)&
\end{aligned}$$

Moreover, if either one of S and O is a variable and the other is a non-variable t or both S and O are the same non-variable t, then the following rule is added:

$$ans_i(Id, X, X, D) :- \; not \; term(X), X = t, Id = [].$$

Finally, the negated property path $S \; !P \; O$ allows the exclusion of top-level subjects S and objects O connected via some path specified by P (Definition 6.3.19).

Definition 6.3.19 (Negated Property Path)**.** *Let PP_i be the i-th subexpression of a property path expression PP and furthermore let $PP_i = !(\mathcal{P})$ be a negated property path expression. Then $\tau_{PP}(PP_i,\; false, S, O, D, i)$ is defined as:*

$$\begin{aligned}
ans_i(Id, X, Y, D) &:- \; triple(X, P, Y, D), P \;!= \; p_{f_1}, \ldots, P \;!= \; p_{f_n}.\\
ans_i(Id, Y, X, D) &:- \; triple(X, P, Y, D), P \;!= \; p_{b_1}, \ldots, P \;!= \; p_{b_m}.
\end{aligned}$$

Here, we are using the following notation:

- $p_{f_1}, \ldots, p_{f_n} \in \{p \mid p \in \mathcal{P}\}$ ... *i.e. the set of negated forward predicates.*
- $p_{b_1}, \ldots, p_{b_m} \in \{p \mid \hat{\;} p \in \mathcal{P}\}$ ... *i.e. the set of negated backward predicates.*

6.3.4 Translation of Query Forms

Let P_1 be a graph pattern and W be a set of variables. We consider two types of query forms: (SELECT W P_1) and (ASK P_1). Their translation is given below.

Definition 6.3.20 (Select)**.** *Let P_i be the i-th subpattern of P and furthermore let P_i be* (SELECT W P_1)*, then $\tau(P_i, true, D, i)$ is defined as:*

$$\begin{aligned}
ans_i(\overline{var}(W), D) &:- \; ans_{2i}(\overline{var}(P_1), D).\\
\tau(P_1, true, D, 2i)&
\end{aligned}$$

And $\tau(P_i, \mathit{false}, D, i)$ is defined as:

$$\begin{aligned} \mathit{ans}_i(\mathit{Id}, \overline{\mathit{var}}(W), D) &:-\ \mathit{ans}_{2i}(\mathit{Id}_1, \overline{\mathit{var}}(P_1), D). \\ &\tau(P_1, \mathit{false}, D, 2i) \end{aligned}$$

Definition 6.3.21 (Ask)**.** *Let P_i be the i-th subpattern of P and furthermore let P_i be* ASK P_1)*, then $\tau(P_i, \mathit{true}, D, i)$ is defined as:*

$$\begin{aligned} \mathit{ans}_i(\mathit{HasResult}) &:-\ \mathit{ans\_ask}_i(\mathit{HasResult}). \\ \mathit{ans}_i(\mathit{HasResult}) &:-\ \mathit{not}\ \ \mathit{ans\_ask}_i(\mathit{true}), \mathit{HasResult} = \mathit{false}. \\ \mathit{ans\_ask}_i(\mathit{HasResult}) &:-\ \mathit{ans}_{2i}(\overline{\mathit{var}}(P_1), D), \mathit{HasResult} = \mathit{true}. \\ &\tau(P_1, \mathit{true}, D, 2i) \end{aligned}$$

And $\tau(P_i, \mathit{false}, D, i)$ is defined as:

$$\begin{aligned} \mathit{ans}_i(\mathit{HasResult}) &:-\ \mathit{ans\_ask}_i(\mathit{HasResult}). \\ \mathit{ans}_i(\mathit{HasResult}) &:-\ \mathit{not}\ \ \mathit{ans\_ask}_i(\mathit{true}), \mathit{HasResult} = \mathit{false}. \\ \mathit{ans\_ask}_i(\mathit{HasResult}) &:-\ \mathit{ans}_{2i}(\mathit{Id}_1, \overline{\mathit{var}}(P_1), D), \mathit{HasResult} = \mathit{true}. \\ &\tau(P_1, \mathit{false}, D, 2i) \end{aligned}$$

6.4 Correctness of the Translation

To ensure the correctness of our translation, we have applied a two-way strategy, consisting of an extensive empirical evaluation and a formal analysis. For the empirical evaluation, we have run our SparqLog system, as well as Fuseki and Virtuoso, on several benchmarks, which provide a good coverage of SPARQL 1.1. The results of our empirical evaluation are summarized in Section 6.6.2. In a nutshell, SparqLog and Fuseki turn out to fully comply with the SPARQL 1.1 standard, while Virtuoso shows deviations from the standard on quite some queries. To provide yet further evidence, we will now formally examine the Warded Datalog$^\pm$ rules produced by our translation for the various SPARQL language constructs and compare them with the formal semantics of these language constructs.

As was mentioned in Section 6.2, SparqLog includes a translation engine with three methods, namely a (*i*) data translation, (*ii*) query translation, and (*iii*) solution translation method. The data translation is very straightforward. In particular, the IRIs, literals, and blank nodes, as well as the triples in an RDF graph, are presented as Datalog ground facts in the obvious way. Recall from Table 6.1 that, as far as query forms are concerned, we currently only support SELECT (which is by far the most common one) and ASK. The former allows one to define a projection to some of the variables in the graph pattern, while the latter just asks if at least some mapping satisfying the graph pattern exists. In the case of SELECT, the solution modifier can be further extended by a DISTINCT, ORDER BY, LIMIT, or OFFSET clause. The two supported solution modifiers (with the possible extensions) are obvious and they are taken care of by the solution translation method of SparqLog. In the following, we restrict our discussion to the query translation method. We treat the basic translation rules and the translation of property paths in separate subsections.

6.4.1 Basic Translation Rules

First, we recall some basic principles of defining a formal semantics of SPARQL (Angles and Gutierrez, 2008; Arenas et al., 2009; Polleres and Wallner, 2013). At the heart of evaluating a SPARQL query is the evaluation of the graph pattern (GP) given in the WHERE clause of the query. This evaluation is relative to the active graph D, which is initially the default graph (obtained by merging the graphs given in the FROM clause of the query) and which can be switched to some named graph (given by an IRI in a FROM NAMED clause of the query) via the GRAPH construct. We write $\mathrm{gr}(u)$ to denote the graph with name u and we write *names* to denote all names of named graphs according to the FROM NAMED clauses.

The result of evaluating a graph pattern P relative to some graph D, denoted by $[\![P]\!]_D$, is a multiset of partial mappings $\mu\colon V \to T$ (simply referred to as "mappings" henceforth), where V is the set of variables and T is the set of terms (i.e., the union of IRIs, blank nodes, and literals). It is convenient to allow also the constant "null" as function value to indicate by $\mu(?X) =$ "null" that μ is undefined on variable $?X$. The domain of μ, denoted $\mathrm{dom}(\mu)$, is defined as the set of variables on which μ is defined. Mappings are applied to triple patterns in the obvious way, i.e., let $t = (s, p, o)$ be a triple pattern and let $\mathrm{var}(t)$ denote the variables in t. For a mapping μ with $\mathrm{var}(t) \subseteq \mathrm{dom}(\mu)$, we write $\mu(t)$ to denote the triple obtained by replacing each variable $?X \in \mathrm{var}(t)$ by $\mu(?X)$.

Compatibility. An important property when combining or comparing two mappings is compatibility. Two mappings μ_1, μ_2 are *compatible*, denoted $\mu_1 \sim \mu_2$, if $\mu_1(?X) = \mu_2(?X)$ holds for all $?X \in \mathrm{dom}(\mu_1) \cap \mathrm{dom}(\mu_2)$. In this case, the mapping $\mu = \mu_1 \cup \mu_2$ with $\mu(?X) = \mu_1(?X)$ if $?X \in \mathrm{dom}(\mu_1)$ and $\mu(?X) = \mu_2(?X)$ if $?X \in \mathrm{dom}(\mu_2)$ is well-defined.

In Section 6.3.2, we have also defined the compatibility of two individual terms or nulls v_1, v_2, namely: v_1 and v_2 are compatible if they are equal (i.e., either the same term or both 'null") or if one of them is "null". Clearly, two partial mappings μ_1, μ_2 are compatible if and only if $\mu_1(?X)$ and $\mu_2(?X)$ are compatible for every variable $?X \in \mathrm{var}(\mu_1) \cap \mathrm{var}(\mu_2)$. If this is the case, then $\mu = \mu_1 \cup \mu_2$ is obtained as follows: for every variable $?X$, (1) if $\mu_1(?X) = \mu_2(?X)$ (where $\mu_1(?X)$ and $\mu_2(?X)$ are either the same term or they are both "null"), then $\mu(?X) = \mu_1(?X) = \mu_2(?X)$; and (2) if one of $\mu_1(?X), \mu_2(?X)$ is a term and the other is "null", then $\mu(?X)$ is set equal to the term.

We observe that the auxiliary predicate $\mathit{comp}(X_1, X_2, X_3)$ defined in Section 6.3.2 realizes precisely the compatibility check between two values $\mu_1(?X)$ and $\mu_2(?X)$ (in the first two components of *comp*) and yields $\mu(?X)$ in the third component.

Operations on Multisets of Mappings. We consider the following operations between two sets of mappings Ω_1, Ω_2:

Graph pattern P	Evaluation $[\![P]\!]_D$
(GRAPH $u\, P_1$)	$[\![P_1]\!]_{\mathrm{gr}(u)}$ if $u \in \mathit{names}$ and $\emptyset$ otherwise
(GRAPH $?X\, P_1$)	$\bigcup_{v \in \mathit{names}}([\![P_1]\!]_{\mathrm{gr}(v)} \bowtie \{?X \to v\})$
$(P_1 \,.\, P_2)$	$[\![P_1]\!]_D \bowtie [\![P_2]\!]_D$
(P_1 FILTER C)	$\{\mu \mid \mu \in [\![P_1]\!]_D$ and $\mu \models C\}$
(P_1 OPT P_2)	$[\![P_1]\!]_D ⟕ [\![P_2]\!]_D$
(P_1 OPT (P_2 FILTER C))	$\{\{\mu \mid \mu \in [\![P_1]\!]_D \bowtie [\![P_2]\!]_D$ and $\mu \models C\}\} \cup$ $\{\{\mu_1 \mid \mu_1 \in [\![P_1]\!]_D$ and for all $\mu_2 \in [\![P_2]\!]_D$: either $\mu_1 \not\sim \mu_2$ or $\mu_1 \sim \mu_2$ and $\mu_1 \cup \mu_2 \not\models C\}\}$
(P_1 UNION P_2)	$[\![P_1]\!]_D \cup [\![P_2]\!]_D$
(P_1 MINUS P_2)	$\{\{\mu_1 \in [\![P_1]\!]_D \mid$ for all $\mu_2 \in [\![P_2]\!]_D$, $(\mu_1 \not\sim \mu_2$ or $\mathrm{dom}(\mu_1) \cap \mathrm{dom}(\mu_2) = \emptyset)\}\}$

Table 6.2: Semantics of basic graph patterns. P_1, P_2 are graph patterns, C is a filter constraint, $u \in I$ and $?X \in V$.

$$\Omega_1 \bowtie \Omega_2 = \{\{\mu_1 \cup \mu_2 \mid \mu_1 \in \Omega_1, \mu_2 \in \Omega_2 \text{ and } \mu_1 \sim \mu_2\}\}$$
$$\Omega_1 \cup \Omega_2 = \{\{\mu \mid \mu \in \Omega_1 \text{ or } \mu \in \Omega_2\}\}$$
$$\Omega_1 \setminus \Omega_2 = \{\{\mu_1 \in \Omega_1 \mid \text{ for all } \mu_2 \in \Omega_2, \mu_1 \not\sim \mu_2\}\}$$
$$\Omega_1 ⟕ \Omega_2 = (\Omega_1 \bowtie \Omega_2) \cup (\Omega_1 \setminus \Omega_2)$$

Note that, of the above operations, only the union $\cup$ may alter the cardinality of elements in the resulting multiset, namely, if a mapping μ is contained in both Ω_1 and Ω_2, then its cardinality in Ω is the sum of the original cardinalities in Ω_1 and Ω_2.

Semantics of Basic SPARQL Constructs. The semantics $[\![P]\!]_D$ of a graph pattern P is defined recursively on the structure of P. In the base case, P is a triple pattern $P = (s, p, o)$ and $[\![P]\!]_D$ is defined as $[\![P]\!]_D = \{\mu \mid \mathrm{dom}(\mu) = \mathrm{var}(P) \text{ and } \mu(P) \in D\}$. For complex graph patterns P, the semantics definition $[\![P]\!]_D$ is shown in Table 6.2.

Translation of Basic SPARQL Constructs. We are now ready to inspect the translations from SPARQL 1.1 to Warded Datalog$^\pm$ given in Section 6.3.2. In the following, we concentrate on bag semantics as they are the more complex case.

Triple. First, consider the base case of graph patterns, namely a triple pattern $P_i = (s, p, o)$, where each of s, p, o can be a term or a variable. Clearly, the single rule produced by our translation $\tau(P_i, \mathit{false}, D, i)$ in Definition 6.3.2 produces all mappings in $\mathrm{var}(P_i)$ that match (s, p, o) to a triple in the active graph D.

Graph. Suppose that P_i is of the form $P_i = (\text{GRAPH } g\ P_1)$. According to the semantics definition in Table 6.2 we have to distinguish two cases depending on whether g is an IRI or a variable. Moreover, in the former case, we have the two subcases depending on whether the IRI g is the name of some named graph (i.e., it occurs in *names*) or not. It is easy to verify that the single rule produced by our translation $\tau(P_i, \mathit{false}, D, i)$ in Definition 6.3.3 covers exactly these three cases.

If g is an IRI that occurs in *names*, then the body literal $\mathit{named}(g)$ of the Datalog rule will evaluate to true, and the resulting mappings (on the variables $\text{var}(P_1)$), obtained by the body literal $\mathit{ans}_{2i}(\mathit{Id}_1, \overline{\mathit{var}}(P_1), g)$ are precisely the mappings obtained by evaluating the graph pattern P_1 over the graph with name g, i.e., $\text{gr}(g)$. In particular, the head variables $\overline{\mathit{var}}(P_i)$ coincide with the body variables $\overline{\mathit{var}}(P_1)$. Note that the variable Id in the head has the effect that every firing of the rule binds Id to a different labeled null. Hence, if $\mathit{ans}_{2i}(\mathit{Id}_1, \overline{\mathit{var}}(P_1), g)$ yields duplicates (i.e., identical mappings with different bindings of Id_1), then these duplicates are preserved by the corresponding firings of the rule (producing a binding of Id to a different labeled null for each firing of the rule).

The rule also behaves correctly in the other 2 cases: if g is an IRI that does not occur in *names*, then the body literal $\mathit{named}(g)$ of the Datalog rule cannot match, and the rule will never fire, thus producing no mapping at all, which is the correct behavior in this case. Finally, if g is a variable, then the body literal $\mathit{named}(g)$ produces mappings of g to all IRIs in *names* and, for each such binding, $\mathit{ans}_{2i}(\mathit{Id}_1, \overline{\mathit{var}}(P_1), g)$ produces precisely the mappings obtained by evaluating graph pattern P_1 over the graph whose name is the current binding of g. Note that, in this case, the head variables $\overline{\mathit{var}}(P_i)$ consist of the variables in P_1 plus the variable g. Again, it is the correct behavior that the rule produces bindings for this increased variable set.

Join. Suppose that P_i is of the form $P_i = (P_1 \,.\, P_2)$. By expanding the definition of the $\bowtie$-operator into the semantics definition in Table 6.2, we get $[\![P_i]\!]_D = \{\{\mu_1 \cup \mu_2 \mid \mu_1 \in [\![P_1]\!]_D,$ $\mu_1 \in [\![P_2]\!]_D$, and $\mu_1 \sim \mu_2\}\}$, that is, the multiset of those mappings which can be obtained as the union of any two *compatible* mappings $\mu_1 \in [\![P_1]\!]_D$ and $\mu_2 \in [\![P_2]\!]_D$.

The rule produced by our translation $\tau(P_i, \mathit{false}, D, i)$ in Definition 6.3.4 achieves precisely this: the two body atoms $\mathit{ans}_{2i}(\mathit{Id}_1, v_1(\overline{\mathit{var}}(P_1)), D)$ and $\mathit{ans}_{2i+1}(\mathit{Id}_2, v_2(\overline{\mathit{var}}(P_2)), D)$ yield the sets of mappings $[\![P_1]\!]_D$ and $[\![P_2]\!]_D$. Note that the variable renaming functions v_1 and v_2 make sure that there is no interference between the evaluation of $[\![P_1]\!]_D$ (by the first body atom) and the evaluation of $[\![P_2]\!]_D$ (by the second body atom). The *comp*-atoms in the rule's body make sure that μ_1 and μ_2 are compatible on all common variables. Moreover, they bind the common variables $\{x_1, \ldots, x_n\}$ to the correct values according to the definition of the *comp*-predicate. In particular, the *comp*-atoms set the common variables to the same value $x_j = v_1(x_j) = v_2(x_j)$ if $v_1(x_j)$ and $v_2(x_j)$ are bound to the same term or if they are both bound to "null". Otherwise, if one of $v_1(x_j)$, $v_2(x_j)$ is a term and the other is "null", then the *comp*-atoms set x_j equal to the term. Finally, recall that the compatibility of two mappings μ_1, μ_2 is defined as the compatibility of all common variables of the two mappings. Hence, the *comp*-atoms in the rule's body produced by our translation $\tau(P_i, \mathit{false}, D, i)$ indeed verify that the two mappings μ_1, μ_2 are compatible.

Filter. Suppose that P_i is of the form $P_i = (P_1 \,\text{FILTER}\, C)$. By the semantics definition in Table 6.2, $[\![P_i]\!]_D$ contains those mappings μ of $[\![P_1]\!]_D$ which satisfy the filter condition C. This is precisely what the single rule resulting from our translation $\tau(P_i, \mathit{false}, D, i)$ in Definition 6.3.7 achieves: the body atom $\mathit{ans}_{2i}(\overline{\mathit{var}}(P_1), D)$ yields all those variable bindings that correspond to the mappings in $[\![P_1]\!]_D$; and adding the filter condition C to the body of the rule means that the rule only fires for variable bindings (strictly speaking, for the mappings corresponding to these variable bindings) for which condition C evaluates to true.

Optional. Suppose that P_i is of the form $P_i = (P_1 \,\text{OPT}\, P_2)$. By expanding the definition of the $⟕$-operator into the semantics definition in Table 6.2, we get $[\![P_i]\!]_D = ([\![P_1]\!]_D \bowtie [\![P_2]\!]_D) \cup ([\![P_1]\!]_D \setminus [\![P_2]\!]_D)$. The translation $\tau(P_i, \mathit{false}, D, i)$ in Definition 6.3.6 yields three rules. The second rule is identical to the translation of a Join expression. As was argued above, it computes precisely the variable bindings corresponding to the mappings in $[\![P_1]\!]_D \bowtie [\![P_2]\!]_D$.

It remains to show that the first and third rule taken together produce the variable bindings corresponding to the mappings in $[\![P_1]\!]_D \setminus [\![P_2]\!]_D$. The first rule is almost the same as the second one, with the only difference being that it projects the join result to the variables in $\text{var}(P_1)$. In other words, it determines the variable bindings corresponding to the mappings in $[\![P_1]\!]_D$, which are compatible with some mapping in $[\![P_2]\!]_D$. Therefore, the first two body literals of the third rule have the following effect: the first literal produces all variable bindings corresponding to mappings in $[\![P_1]\!]_D$ while the second (i.e., the negative) body literal selects those variable bindings which correspond to mappings that are *not compatible* with any mapping in $[\![P_2]\!]_D$. By setting all variables in $\text{var}(P_2) \setminus \text{var}(P_1)$ to "null" (with the remaining m body atoms), the third rule indeed produces the variable bindings corresponding to the mappings in $[\![P_1]\!]_D \setminus [\![P_2]\!]_D$.

Optional Filter. Suppose that P_i is an optional filter expression of the form $P_i = (P_1 \,\text{OPT}(P_2 \,\text{FILTER}\, C))$. According to the semantics definition in Table 6.2, $[\![P_i]\!]_D$ is obtained as the union of 2 multisets:

1. the mappings μ in $[\![(P_1 \,.\, P_2)]\!]_D$ which satisfy the filter condition C;
2. the mappings μ_1 in $[\![P_1]\!]_D$ for which all mappings μ_2 in $[\![P_2]\!]_D$ have one of the following two properties: either μ_1 and μ_2 are not compatible or they are compatible, but their combination does not satisfy the filter condition C.

The translation $\tau(P_i, \mathit{false}, D, i)$ in Definition 6.3.8 yields three rules, which are very similar to the translation of Optional expressions discussed before. The only difference is that now, the first and second rules have filter condition C as additional body literals. Compared with the rules in the case of Optional expressions, these additional body literals have the following effect:

- The second rule computes the variable bindings corresponding to those mappings in $[\![P_1]\!]_D \bowtie [\![P_2]\!]_D$ which satisfy the filter condition C. That is, the mappings according to item 1 above.
- The first rule computes those variables bindings corresponding to the mappings in $[\![P_1]\!]_D$ which are compatible with some mapping in $[\![P_2]\!]_D$ *and* which, together with a compatible mapping from $[\![P_2]\!]_D$ satisfy the condition C.

Therefore, the (negative) second body literal in the third rule has the effect of eliminating precisely those mappings μ_1 from the multiset of mappings in $[\![P_1]\!]_D$ (obtained via the first body atom) for which there exists a compatible mapping μ_2 in $[\![P_2]\!]_D$, such that their combination satisfies the filter condition C. In other words, we are left with the mappings from item 2 above. Analogously to Optional patterns, the variables in $\text{var}(P_2) \setminus \text{var}(P_1)$ are not part of the domain of these mappings. Hence, with the *null*-atoms in the body of the third rule, we set all these variables to "null".

Union. Suppose that P_i is of the form $P_i = (P_1 \,\text{UNION}\, P_2)$. According to the semantics definition in Table 6.2, $[\![P_i]\!]_D$ is simply obtained as the union of the two multisets $[\![P_1]\!]_D$ and $[\![P_2]\!]_D$. In principle, the two rules of our translation $\tau(P_i, \mathit{false}, D, i)$ in Definition 6.3.5 compute this union of the variable bindings corresponding to the mappings in $[\![P_1]\!]_D$ (via the body atom $\mathit{ans}_{2i}(\mathit{Id}_1, \overline{\mathit{var}}(P_1), D)$ in the first rule) and the variable bindings corresponding to the mappings in $[\![P_2]\!]_D$ (via the body atom $\mathit{ans}_{2i+1}(\mathit{Id}_2, \overline{\mathit{var}}(P_2), D)$ in the second rule). However, care has to be taken that all variable bindings obtained for $\mathit{ans}_i(\mathit{Id}, \overline{\mathit{var}}(P_i), D)$ must be defined on *all* variables in $\text{var}(P_i)$. Therefore, variable bindings obtained from $\mathit{ans}_{2i}(\mathit{Id}_1, \overline{\mathit{var}}(P_1), D)$ have to be extended to the variables in $\text{var}(P_2) \setminus \text{var}(P_1)$ by setting the latter explicitly to "null". Likewise, the variable bindings obtained from $\mathit{ans}_{2i+1}(\mathit{Id}_1, \overline{\mathit{var}}(P_2), D)$ have to be extended to the variables in $\text{var}(P_1) \setminus \text{var}(P_2)$ by setting the latter explicitly to "null". This is achieved by the *null*-atoms in the rule bodies of the two rules.

Minus. Suppose that P_i is of the form $P_i = (P_1 \,\text{MINUS}\, P_2)$. According to the semantics definition in Table 6.2, $[\![P_i]\!]_D$ consists of those mappings μ_1 of $[\![P_1]\!]_D$ which, for any mapping μ_2 of $[\![P_2]\!]_D$ satisfy one of the following two conditions: either μ_1 and μ_2 are not compatible or $\text{dom}(\mu_1)$ and $\text{dom}(\mu_2)$ have no variable in common. In other words, a mapping $\mu_1 \in [\![P_1]\!]_D$ is retained in $[\![P_i]\!]_D$ unless there exists a mapping $\mu_2 \in [\![P_2]\!]_D$ such that μ_1 and μ_2 are compatible, and there exists at least one variable x with $\mu_1(x) = \mu_2(x) \neq$"null".

Similar to our translation of Join patterns, the first rule of our translation $\tau(P_i, \mathit{false}, D, i)$ of a Minus pattern in Definition 6.3.9 computes the variable bindings of the variables in $\text{var}(P_1)$ and of $\text{var}(P_2)$ which correspond to compatible mappings. The next n rules (all with head predicate $\mathit{ans}_{\mathit{equal}-i}$) restrict the set of compatible mappings to those whose domains have at least one variable in common, i.e., the corresponding variable bindings have at least one variable on which they coincide, and they are both not "null". Note that the signature of $\mathit{ans}_{\mathit{equal}-i}$ is restricted to the variables in $\text{var}(P_1)$. That is, $\mathit{ans}_{\mathit{equal}-i}$

contains all variable bindings on var(P_1), which correspond to “forbidden” mappings. The last rule in our translation $\tau(P_i, \mathit{false}, D, i)$ computes the variable bindings corresponding to the mappings in $[\![P_i]\!]_D$ by computing the variable bindings corresponding to the mappings in $[\![P_1]\!]_D$ (via the first body literal) and eliminating the “forbidden” ones (via the negative second body literal).

6.4.2 Translation Rules for Property Paths

Analogously to the previous section, we now also juxtapose the semantics of property path expressions with our translation.

Semantics of Property Paths. For a property path PP, we write $[\![PP]\!]_{D,s,o}$ to denote the semantics of a property path PP over a graph D with s, o denoting the subject and object of the top-level property path expression. The semantics of a property path PP is a pair (x, y) of terms such that there is a path PP from x to y. Here, we mainly follow the semantics definition in (Kostylev et al., 2015). The semantics of property paths is defined recursively, with link property paths (i.e., simply an IRI p) as the base case.

The definition $[\![PP]\!]_{D,s,o}$ for arbitrary property paths PP is given in Table 6.3. There, we write *Distinct* for converting a multiset into a set by deleting duplicates. Moreover, we write $\mathit{reach}(x, PP, D, s, o)$ for the set of terms reachable from some start point x by applying the path PP one or more times, where s, o are again the subject and object of the top-level property path expression.

Recall from the previous section that the semantics $[\![P]\!]_D$ of a graph pattern P over a graph D is defined as a multiset of (partial) mappings. If a graph pattern P is of the form $P = (s\, PP\, o)$, where PP is a property path, then $[\![(s\, PP\, o)]\!]_D$ is the multiset of mappings obtained as follows:

$$\begin{aligned}[\![s\, PP\, o]\!]_D = \{\!\{\mu \mid \mathit{dom}(\mu) = \mathrm{var}(\{s, o\}) \text{ and}\\ (\mu(s), \mu(o)) \in [\![PP]\!]_{D,s,o}\}\!\},\end{aligned}$$

where we write $\mu(x)$ with $x \in \{s, o\}$ for both variables and non-variables x with the understanding that $\mu(x) = x$ if $x \notin V$.

Graph Pattern with a Property Path. Before we inspect the translations of property paths to Warded Datalog$^\pm$, we inspect our translation of graph patterns *using* a property path. That is, consider a graph pattern P of the form $P = s\, PP_1\, o$, where PP_1 is a property path. We have recalled above the definition of $[\![s\, PP_1\, o]\!]_D$ as a multiset of pairs of terms. Now suppose that our translation $\tau_{PP}(PP_1, \mathit{false}, S, O, D, 2i)$ of property path PP_1 is correct. (A proof sketch of this fact comes next.) Then, the single additional rule of our translation in Definition 6.3.10, with head atom $\mathit{ans}_{2i}(\mathit{Id}_1, S, O, D)$, indeed produces all mappings μ on $\mathit{var}(P)$ such that $(\mu(S), \mu(O))$ is in $[\![PP_1]\!]_{D,\mu(S),\mu(O)}$. Note that the multiplicities of the mappings thus obtained are taken care of by different bindings of the variable Id_1.

Property Path PP	Evaluation $[\![PP]\!]_{D,s,o}$
p	$\{\{(x,y) \mid (x,p,y) \in D\}\}$
$\hat{\ } PP_1$	$\{\{(x,y) \mid (y,x) \in [\![PP]\!]_{D,s,o}\}\}$
$PP_1 \mid PP_2$	$[\![PP_1]\!]_{D,s,o} \cup [\![PP_2]\!]_{D,s,o}$
PP_1/PP_2	$\{\{(x,z) \mid \exists y\colon (x,y) \in [\![PP_1]\!]_{D,s,o}$ $\wedge (y,z) \in [\![PP_2]\!]_{D,s,o}\}\}$
PP_1+	$\{(x,y) \mid y \in reach(x, PP_1, D, s, o)\}$
$PP_1?$	$Distinct([\![PP_1]\!]_{D,s,o})$ $\cup \{(x,x) \mid (x,y,z) \in D\}$ $\cup \{(z,z) \mid (x,y,z) \in D\}$ $\cup \{(s,s) \mid s \notin V \wedge o \in V\}$ $\cup \{(o,o) \mid o \notin V \wedge s \in V\}$ $\cup \{(s,s) \mid s \notin V \wedge s = o\})$
PP_1*	$Distinct(PP_1? \cup PP_1+)$
$!\mathcal{P}$ with $p_{f_1}, \ldots, p_{f_n} \in \{p \mid p \in \mathcal{P}\} \wedge$ $p_{b_1}, \ldots, p_{b_m} \in \{p \mid \hat{\ } p \in \mathcal{P}\}$	$\{\{(x,y) \mid \exists a\colon (x,a,y) \in D$, s.t. $a \notin \{p_{f_1}, \ldots, p_{f_n}\}\}\} \cup$ $\{\{(x,y) \mid \exists a\colon (y,a,x) \in D$, s.t. $a \notin \{p_{b_1}, \ldots, p_{b_m}\}\}\}$

Table 6.3: Semantics of SPARQL property paths. PP_1 and PP_2 are property paths; $\mathcal{P}$ is a set of link property paths and inverse link property paths; $p \in I$; and s and o are the subject and object of the top-level property path expression.

Translation of Property Paths. We are now ready to inspect the translations of property paths to Warded Datalog$^{\pm}$ given in Section 6.3.3. Again, we concentrate on bag semantics as the more complex case. We proceed inductively on the structure of the property paths. In all cases, suppose that we want to evaluate property paths over a graph D and let s, o denote the top-level subject and object, given in a graph pattern of the form $(s\, PP\, o)$. We start with link property paths as the base case and then cover all types of compound property path expressions.

Link Property Path. Suppose that the property path PP_i consists of a single IRI p, i.e., $PP_i = p$. Then $[\![PP]\!]_{D,s,o}$ consists of all pairs (x,y), such that (x,p,y) is a triple in D. On the other hand, the single rule produced by our translation $\tau_{PP}(PP_i, \mathit{false}, S, O, D, i)$ in Definition 6.3.11 yields exactly these pairs of terms.

Inverse Path. Consider a property path PP_i of the form $PP_i = \hat{\ } PP_1$ for some property path PP_1. Then $[\![PP_i]\!]_{D,s,o}$ consists of all pairs (y,x) such that (x,y) is contained in $[\![PP_1]\!]_{D,s,o}$. That is, $[\![PP_i]\!]_{D,s,o}$ just swaps first and second component of each pair in

$[\![PP_1]\!]_{D,s,o}$. This is exactly what the single rule in our translation $\tau_{PP}(PP_i, \mathit{true}, S, O, D, i)$ in Definition 6.3.12 does.

Alternative Path. Consider a property path PP_i of the form $PP_i = PP_1 \mid PP_2$ for some property paths PP_1 and PP_2. Then, according to the semantics definition in Table 6.3, $[\![PP_i]\!]_{D,s,o}$ consists of the union of $[\![PP_1]\!]_{D,s,o}$ and $[\![PP_2]\!]_{D,s,o}$. The two rules in our translation $\tau_{PP}(PP_i, \mathit{true}, S, O, D, i)$ in Definition 6.3.13 realize exactly this union.

Sequence Path. Consider a property path PP_i of the form $PP_i = PP_1/PP_2$ for some property paths PP_1 and PP_2. Then, according to the semantics definition in Table 6.3, $[\![PP_i]\!]_{D,s,o}$ consists of pairs (x, z) (i.e., start and end points of paths described by PP_i) such that there exist pairs $(x, y) \in [\![PP_1]\!]_{D,s,o}$ and $(y, z) \in [\![PP_2]\!]_{D,s,o}$ (i.e., start and end points of paths described by PP_1 and PP_2, respectively, such that the end point of a path according to PP_1 and the start point of a path according to PP_2 coincide). The single rule in our translation $\tau_{PP}(PP_i, \mathit{true}, S, O, D, i)$ in Definition 6.3.14 realizes precisely these combinations of pairs $(x, y) \in [\![PP_1]\!]_{D,s,o}$ and $(y, z) \in [\![PP_2]\!]_{D,s,o}$.

One-Or-More Path. Consider a property path PP_i of the form $PP_i = PP_1+$ for some property path PP_1. The semantics of the one-or-more path expression is essentially that of reachability via hops defined by the property path expression PP_1. That is, we get all pairs (x, y) that are in the "infinite" union $[\![PP_1]\!]_{D,s,o} \cup [\![PP_1/PP_1]\!]_{D,s,o} \cup [\![PP_1/PP_1/PP_1]\!]_{D,s,o} \cup [\![PP_1/PP_1/PP_1/PP_1]\!]_{D,s,o} \cup \dots$, with one important difference though: according to the SPARQL semantics of property paths[2], *one-or-more* property paths (and likewise *zero-or-one* and *zero-or-more* property paths) always have set semantics. The two rules in our translation $\tau_{PP}(PP_i, \mathit{true}, S, O, D, i)$ in Definition 6.3.17 realize exactly this kind of reachability relationship. Moreover, neither in the semantics definition nor in the translation, we need to keep track of duplicates. Therefore, in the semantics definition, we define $[\![PP_i]\!]_{D,s,o}$ as a set (rather than a multiset). In our translation, duplicates are avoided by the $Id = []$ body atom in both rules. As was already mentioned in Section 6.3.3, this body atom has the effect that no copies of the same pair (x, y) (but with different binding of Id) can ever be produced.

Zero-Or-One Path. Consider a property path PP_i of the form $PP_i = PP_1?$ for some property path PP_1. Then, intuitively, $[\![PP_i]\!]_{D,s,o}$ consists of pairs of nodes that are the start and end point of a "one-path" (i.e., traversing PP_1 once) plus "zero-paths" (i.e., identical start and end points). In the semantics definition in Table 6.3, the pairs corresponding to "one-paths" are taken care of by the expression $[\![PP_1]\!]_{D,s,o}$. Analogously, in our translation $\tau_{PP}(PP_i, \mathit{true}, S, O, D, i)$ in Definition 6.3.16 these pairs are produced by the second rule.

All remaining expressions in the semantics definition correspond to various ways of getting "zero-paths", namely either for every term in the graph (captured by the second and third expression of the semantics definition) or if at least one of s or o is a term (captured by the remaining expressions of the semantics definition). In case both s and o

[2]https://www.w3.org/TR/SPARQL11-query/#defn_PropertyPathExpr (last visited 09/25/2023)

are terms, they must be the same in order to constitute a "zero-path". In our translation $\tau_{PP}(PP_i, true, S, O, D, i)$ in Definition 6.3.16, the first rule produces the pairs (x, x) for terms x occurring in the active graph. The last rule of the translation produces the remaining pairs (t, t) if t is a term that occurs as a top-level subject or object of the entire property path expression. As with one-or-more path expressions, it is important to keep in mind that also zero-or-one paths always have set semantics according to the SPARQL semantics of property paths. The elimination of duplicates is ensured by the *Distinct* operator in the semantics definition and by the $Id = []$ body atom in the rules of our translation.

Zero-Or-More Path. Consider a property path PP_i of the form $PP_i = PP_1*$ for some property path PP_1. In our semantics definition in Table 6.3, we have defined $[\![PP_i]\!]_{D,s,o}$ simply as the set-variant (i.e., deleting any duplicates) of the union of the zero-or-one path and the one-or-more path. Of course, in the case of bag semantics this would be problematic since we thus count "one-paths" twice. However, since *zero-or-more* property paths always have set semantics, the *Distinct* operator applied to the union eliminates any duplicates anyway. The rules in our translation $\tau_{PP}(PP_i, true, S, O, D, i)$ in Definition 6.3.18 are indeed obtained as the union of the rules that one gets for the translations of the zero-or-one path $PP_1?$ and of the one-or-more path PP_1+. The $Id = []$ body atom in each of the rules makes sure that we never produce any copies of any pair (x, y).

Negated Path. Consider a property path PP of the form $PP = !\mathcal{P}$, where $\mathcal{P}$ is a set of "forward" link property path expressions $\{p_{f_1}, \ldots, p_{f_n}\}$ and "backward" link property path expressions $\{\hat{}p_{b_1}, \ldots, \hat{}p_{b_m}\}$. Then, according to our semantics definition in Table 6.3, $[\![PP_i]\!]_{D,s,o}$ contains those pairs (x, y) for which either there exists a triple (x, a, y) in the active graph such that a is different from all p_{f_j} or there exists a triple (y, a, x) in the active graph such that a is different from all p_{b_j}. Our translation $\tau_{PP}(PP_i, true, S, O, D, i)$ in Definition 6.3.19 generates two rules. Clearly, the first type of pairs in $[\![PP_i]\!]_{D,s,o}$ is produced by the first rule of our translation, and the second type of pairs in $[\![PP_i]\!]_{D,s,o}$ is produced by the second rule.

6.5 Implementation Details

Based on the translation from SPARQL to Warded Datalog$^{\pm}$, covered in Section 6.3, we implemented the SparqLog system. In this section, we state details of its implementation.

Some Basic Principles. The SPARQL to Warded Datalog$^{\pm}$ translator was implemented in Java using the library org.apache.jena to parse SPARQL query strings and handle operations, solution modifiers, basic graph patterns, etc. appropriately. The ARQ algebra query parser[3] of Apache Jena parses SPARQL query strings in a top-down fashion. First, query forms are parsed; next, solution modifiers; and in the end, operations starting from the outer-most operation going inward.

[3]https://jena.apache.org/documentation/query/ (last visited 09/25/2023)

In contrast, our developed SPARQL to Warded Datalog$^{\pm}$ parser analyzes queries bottom-up. Thus, the translation starts at basic graph patterns and continues upwards. This setup is necessary, as the variables inside an expression need to be kept track of when parsing it, as rules usually modify the results of sub-operations. Therefore, it needs to be known which variables occur in the respective subresult predicates.

Datatypes, Languages, and Compatibility. Our translation engine partially supports datatypes and language tags by adding two additional arguments to each variable containing the respective information. This has implications on most SPARQL operations such as *UNION*, *JOIN*, and *FILTER*. For example, in the case of *JOIN* operations, we have extended the existing translation of (Angles and Gutierrez, 2016a) by developing two additional comparison predicates (*compD* and *compL*). In (Angles and Gutierrez, 2016a), the predicate *comp* is used for computing the compatibility between two variables. The new predicates *compD* and *compL* are used to compute the respective compatibility for their datatypes and language tags, which is done in the same way as for variables thus far.

Moreover, Vadalog (and Datalog in general) joins variables of the same rule by name. However, the semantics of joins in SPARQL differs from that of Datalog. It is for that reason (1) that we have to prefix/rename variables in such a way that Vadalog's internal join strategy is prevented and (2) that we introduce the join predicate *comp* described in Section 6.3.2 to realize SPARQL join semantics.

Skolem Functions (For Bag Semantics). The Skolem function generator lies at the heart of how we preserve duplicate results. As in the work of (Bertossi et al., 2019), we introduce IDs to preserve Datalog bag semantics in Warded Datalog$^{\pm}$ set semantics. Therefore, each generated result tuple is distinguished from its duplicates by a tuple ID (referred to as TID in (Bertossi et al., 2019)). However, instead of simply generating nulls, our ID generation process is abstracted away by the *getSkolF* function of the *skolFG* object. We thus generate IDs as follows. Since the grounding of each positive atom in the rule body is responsible for the generation of a tuple in Datalog$^{\pm}$, we extract a sorted list of all variables occurring in positive atoms of the rule body *bodyVars*. Finally, the tuple ID is generated by assigning it to a list starting with the string "$f_{<ruleID>}$", followed by the list of positive body variables *bodyVars* and a string *label*. The strings "$f_{<ruleID>}$" and *label* were added, as we use "$f_{<ruleID>}$" to identify the translated rules of the processed operator at the current translation step, while *label* provides additional information when needed.

This setup preserves generated duplicates of SPARQL bag semantics in Warded Datalog$^{\pm}$ set semantics by utilizing their provenance information to make them distinguishable. Furthermore, it provides information for debugging/explanation purposes of the reasoning process, as each tuple carries the information which rule and grounding have led to its generation. As an added bonus, this layer of abstraction may be used to adapt different duplicate generation semantics/strategies as might be necessary for different applications by simply exchanging the *skolFG* by any self-implemented solution.

6.6 Experimental Evaluation

In this section, we report on the experimental evaluation of the SparqLog system. We want to give a general understanding of the behavior of SparqLog in the following three areas: (1) we first analyze various benchmarks available in the area to identify **coverage** of SPARQL features and which benchmarks to use subsequently in our evaluation, (2) we analyze the **compliance** of our system with the SPARQL standard using the identified benchmarks, and set this in context with the two state-of-the-art systems Virtuoso and Fuseki, and, finally, (3) we evaluate the **performance** of query execution of SparqLog and compare it with state-of-the-art systems for SPARQL query answering and reasoning over ontologies, respectively. We thus put particular emphasis on property paths and their combination with ontological reasoning.

6.6.1 Benchmark Analysis

In this section, we take a deeper look at current state-of-the-art benchmarks for executing SPARQL queries. Following Saleem et al. (2019), we analyze the following benchmarks: BowlognaBench (Bowlogna) (Demartini et al., 2011), TrainBench (Szárnyas et al., 2018a), Berlin SPARQL Benchmark (BSBM) (Bizer and Schultz, 2009), SP2Bench (Schmidt et al., 2008), Waterloo SPARQL Diversity Test Suite (WatDiv) (Aluç et al., 2014), LDBC Social Network Benchmark Business Intelligence Workload (SNB-BI) (Szárnyas et al., 2018b), LDBC SNB Interactive Workload (SNB-INT) (Erling et al., 2015), FEASIBLE (Saleem et al., 2015), Fishmark (Bail et al., 2012), DBpedia SPARQL Benchmark (DBPSB) (Morsey et al., 2011), and BioBench (Wu et al., 2014). Note that FEASIBLE is not directly a benchmark but a query generator, producing queries from real-world query logs. We employed FEASIBLE on two query logs, provided at FEASIBLE's GitHub repository[4], namely on the Semantic Web Dog Food (SWDF) and on the DBPedia query log. We refer to the resulting FEASIBLE benchmark from SWF as *FEASIBLE (S)* and respectively from DBPedia as *FEASIBLE (D)*.

The aim of this analysis is to identify the set of SPARQL features that are covered by the queries of each benchmark, thereby getting some notion of how exhaustively the benchmarks cover the SPARQL language. To reach this goal, we count the SPARQL features of each benchmark query. Following a similar approach as Saleem et al. (2019), we count each feature once per query in which it occurs, with one exception being the *DISTINCT* feature. As in (Saleem et al., 2019), we count the DISTINCT feature only if it is applied to the entire query. This is also in line with our interest of testing bag and set semantics in combination with different SPARQL features.

Table 6.4 represents the result of our exploration of the SPARQL feature coverage of the considered benchmarks. Particularly heavily used SPARQL features are marked in blue, while missing SPARQL features are marked in orange. The abbreviations of the columns represent the following SPARQL features: DIST[INCT], FILT[ER],

[4]https://github.com/dice-group/feasible (last visited 09/25/2023)

REG[EX], OPT[IONAL], UN[ION], GRA[PH], P[roperty Path] Seq[uential], P[roperty Path] Alt[ernative], GRO[UP BY].

	Benchmark	DIST	FILT	REG	OPT	UN	GRA	PSeq	PAlt	GRO
Synthetic	Bowlogna	5.9	41.2	11.8	0.0	0.0	0.0	0.0	0.0	76.5
	TrainBench	0.0	41.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	BSBM	25.0	37.5	0.0	54.2	8.3	0.0	0.0	0.0	0.0
	SP2Bench	35.3	58.8	0.0	17.6	17.6	0.0	0.0	0.0	0.0
	WatDiv	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	SNB-BI	0.0	66.7	0.0	45.8	20.8	0.0	16.7	0.0	100.0
	SNB-INT	0.0	47.4	0.0	31.6	15.8	0.0	5.3	10.5	42.1
Real	FEASIBLE (D)	56.0	58.0	14.0	28.0	40.0	0.0	0.0	0.0	0.0
	FEASIBLE (S)	56.0	27.0	9.0	32.0	34.0	10.0	0.0	0.0	25.0
	Fishmark	0.0	0.0	0.0	9.1	0.0	0.0	0.0	0.0	0.0
	DBPSB	100.0	44.0	4.0	32.0	36.0	0.0	0.0	0.0	0.0
	BioBench	39.3	32.1	14.3	10.7	17.9	0.0	0.0	0.0	10.7

Table 6.4: Feature coverage of SPARQL benchmarks (Saleem et al., 2019)

In contrast to (Saleem et al., 2019), we do not limit our benchmark analysis to *SELECT* queries, but rather analyze all queries provided at the GitHub repository of the dice group[5]. Therefore, we also analyze the *DESCRIBE* queries of, e.g., the BSBM benchmark. Moreover, since the query file of the SP2Bench benchmark does not contain the hand-crafted *ASK* queries provided on its homepage[6], we have chosen to add these to the benchmark to be able to analyze the complete query-set of SP2Bench. For these reasons, the results of overlapping SPARQL features and benchmarks from our analysis in Table 6.4 and the one of (Saleem et al., 2019) differ slightly.

Furthermore, note that we do not display basic features, such as *Join*, *Basic Graph Pattern*, etc. in Table 6.4, as these features are, of course, covered by each of the considered benchmarks. Moreover, we have chosen to only include the *REGEX* filter constraint in the feature coverage table and no other specific constraints, as the *REGEX* function is argued to be of vital importance for SPARQL users in (Saleem et al., 2019). For this reason, we have chosen to cover this feature with our translation engine in addition to the other filter constraints. Finally, we have not included the SPARQL features *MINUS* and the *inverted*, *zero-or-one*, *zero-or-more*, *one-or-more* and *negated property path*, as none of the selected benchmarks covers any of these SPARQL features.

Table 6.4 reveals that no benchmark covers all SPARQL features. Even more, SNB-BI and SNB-INT are the only benchmarks that contain property paths. Yet, they cover merely the *sequential* (PSeq) and *alternative property path* (PAlt), which in principle correspond to the *JOIN* and *UNION* operator. This means that no existing benchmark covers recursive property paths (though we will talk about the benchmark generator gMark (Bagan et al., 2017) later), which are one of the most significant extensions

[5] https://hobbitdata.informatik.uni-leipzig.de/benchmarks-data/queries/ (last visited 09/25/2023)
[6] http://dbis.informatik.uni-freiburg.de/index.php?project=SP2B/queries.php (last visit: 09/25/2023)

provided by SPARQL 1.1. Our analysis of SPARQL benchmarks leads us to the following conclusions for testing the compliance with the SPARQL standard and for planning the performance tests with SparqLog and state-of-the-art systems.

Evaluating Compliance With the SPARQL Standard. Based on the results of Table 6.4, we have chosen the following three benchmarks to evaluate the compliance of our SparqLog system with the SPARQL standard: (1) We have identified *FEASIBLE (S)* (Saleem et al., 2015) as the real-world benchmark of choice, as it produces the most diverse test cases (Saleem et al., 2019) and covers the highest amount of features; (2) *SP2Bench* (Schmidt et al., 2008) is identified as the synthetic benchmark of choice since it produces synthetic datasets with the most realistic characteristics (Saleem et al., 2019); (3) finally since no benchmark that employs real-world settings provides satisfactory coverage of property paths, we have additionally chosen *BeSEPPI* (Skubella et al., 2019) – a simplistic, yet very extensive benchmark specifically designed for testing the correct and complete processing of property paths. We report on the results of testing the compliance of our SparqLog system as well as Fuseki and Virtuoso in Section 6.6.2.

Performance Benchmarking. For the empirical evaluation of query execution times reported in Section 6.6.3, we have identified SP2Bench as the most suitable benchmark, as it contains hand-crafted queries that were specifically designed to target query optimization. Since none of the existing benchmarks for SPARQL performance measurements contains recursive property paths, we have included instances generated by the benchmark generator gMark (Bagan et al., 2017) and report extensive results of this important aspect. In order to include in our tests also the performance measurements for the combination of property paths with ontologies, we have further extended SP2Bench with an ontology containing subPropertyOf and subClassOf statements and report the results in Section 6.6.4.

6.6.2 SPARQL Compliance Tests

As detailed in Section 6.6.1, we have selected three benchmarks (BeSEPPI, SP2Bench and FEASIBLE (S)) for evaluating the standard-compliance of the chosen three systems (SparqLog, Jena Fuseki, and OpenLink Virtuoso). For the SPARQL compliance tests, we use Apache Jena Fuseki 3.15.0 and Virtuoso Open Source Edition, version 7.2.5. The tests are run on a Windows 10 machine with 8GB of main memory. In this section, we explain the setup of the standard-compliance tests that we performed on our SparqLog system and the two state-of-the-art SPARQL engines Fuseki and Virtuoso. Moreover, we mention some challenges with these tests and provide further details on the outcome of the compliance tests.

Benchmark Generation

In order to carry out our standard-compliance tests, we first have to make the queries and the data provided by the benchmarks accessible to the tested systems. While this turns out to be an easy task for BeSEPPI and SP2Bench, some care is required for the FEASIBLE (S) benchmark.

BeSEPPI and SP2Bench. The BeSEPPI benchmark contains queries and a dataset for the evaluation of property path queries. Its dataset can be directly loaded into the selected systems and its queries can be directly executed. The SP2Bench benchmark contains 17 hand-crafted queries and a benchmark dataset generator. For the purpose of our compliance tests, we have generated a dataset with 50k triples, which was loaded into each of the considered systems.

FEASIBLE. The FEASIBLE benchmark contains a query generator, which generates queries for an arbitrary dataset that provides a query log. In the case of the FEASIBLE (S) benchmark, we have chosen the Semantic Web Dog Food (SWDF) dataset and generated 100 queries using the SWDF query log. However, some additional work was required before we could use the FEASIBLE (S) benchmark for our tests.

The first complication arises from the fact that Vadalog uses Java sorting semantics, whereas the SPARQL standard defines its own ordering semantics. We had to remove *LIMIT* and *OFFSET* from each query of the FEASIBLE (S) benchmark, as queries with these features can only be reasonably evaluated (comparing the generated query results, rather than only checking if their cardinalities are equal) if the results are sorted and if each considered RDF query and storage system provides the same sorting semantics. Some queries of the generated benchmark only differed from each other in the argument of the *LIMIT* or *OFFSET* clause. Thus, after removing all *LIMIT* and *OFFSET* clauses, we ended up with duplicate queries. These duplicate queries were eliminated, leaving the FEASIBLE (S) benchmark with a total number of 77 unique queries.

Moreover, Vadalog does currently not support UTF-8 characters. We were, therefore, faced with the necessity of changing the encoding of the SWDF dataset of the FEASIBLE (S) benchmark. We have made the plausible assumption that dropping non-ASCII characters from RDF strings would not lead to vastly different results and we have, therefore, simply deleted all non-ASCII characters from the SWDF dataset.

Furthermore, since the FEASIBLE (S) benchmark includes queries with the *GRAPH* feature (which selects the graph IRI of RDF triples), we have loaded the SWDF dataset both into the default graph of each tested system and into a named graph for the FEASIBLE benchmark to be able to test the GRAPH feature.

Challenges of the Evaluation Process

The evaluation of the standard compliance of the three chosen systems requires the comparison of query results. In the case of BeSEPPI, the benchmark also provides the expected result for each query. We, therefore, have to compare the results produced by each of the three systems with the correct result defined by the benchmark itself. In contrast, FEASIBLE and SP2Bench do not provide the expected results for their queries. We, therefore, use a majority voting approach to determine the correct answer. That is, we compare the query results produced by the three considered systems and accept a result as the expected query answer if it is equal to the generated query result of at least two of the tested systems.

A major challenge for comparing query results (both when comparing the result produced by one system with the expected result defined by the benchmark itself and when comparing the results produced by two systems) comes from *blank nodes*. On the one hand, each system employs its own specific functionality for assigning blank node names. Therefore, to compare blank nodes between the different result multisets, a mapping between the internal system-specific blank node names has to be found. However, finding such a mapping comes down to finding an isomorphism between two arbitrarily sized tables containing only blank nodes, which requires exponential time in the worst case and poses a severe problem for large result multisets with many blank nodes. We have, therefore, tried out a simple heuristic to find a suitable mapping between blank nodes by sorting the query results without considering blank node names first. We then iterate over both results, and finally, each time a new blank node name is encountered, we save the mapping between the system-specific blank node names. Even though this is a very simple heuristic, it has worked quite well in many cases. Nevertheless, there are cases where this simple procedure infers wrong blank node mappings, even though the results are semantically equivalent. Hence, due to the instability of this efficient blank node checking heuristic, we have chosen to remove the evaluation of blank nodes from our compliance tests. That is, our current evaluation test suite does, for this reason, not distinguish between different blank node names, but, of course, it distinguishes between all other terms.

Outcome of the Compliance Tests

FEASIBLE (S). The FEASIBLE(S) benchmark contains 77 queries that we used for testing the standard-conformant behavior. It turned out that both SparqLog and Fuseki fully comply with the standard on each of the 77 queries, whereas Virtuoso does not. More specifically, for 14 queries, Virtuoso returned an erroneous result by either wrongly outputting duplicates (e.g., ignoring DISTINCTs) or omitting duplicates (e.g., by handling UNIONs incorrectly). Moreover, in 18 cases, Virtuoso was unable to evaluate the query and produced an error.

SP2Bench. The SP2Bench benchmark contains 17 queries in total and is specifically designed to test the scalability of SPARQL engines. All three considered systems produce identical results for each of the 17 queries.

BeSEPPI. The BeSEPPI benchmark contains 236 queries specifically designed to evaluate the correct and complete support of property path features. Table 6.5 shows the detailed results of the experimental evaluation of the three considered systems on this benchmark. We distinguish four types of erroneous behavior: *correct but incomplete* results (i.e., the mappings returned are correct, but there are further correct mappings missing), *complete but incorrect* (i.e., no correct mapping is missing, but the answer falsely contains additional mappings), *incomplete and incorrect*, or failing to evaluate the query and returning an *error* instead. The entries in the table indicate the number of cases for each error type. We see that Fuseki and SparqLog produce the correct result in all 236 cases. Virtuoso only handles the queries with inverse, sequence, and negated path expressions 100% correctly. For queries containing alternative, zero-or-one, one-or-more, or zero-or-more path expressions, Virtuoso is not guaranteed to produce the correct result. The precise number of queries handled erroneously is shown in the cells marked red.

Stores	Virtuoso				Jena Fuseki				SparqLog				
Expressions	Incomp. & Correct	Complete & Incor.	Incomp. & Incor.	Error	Incomp. & Correct	Complete & Incor.	Incomp. & Incor.	Error	Incomp. & Correct	Complete & Incor.	Incomp. & Incor.	Error	Total #Queries
Inverse	0	0	0	0	0	0	0	0	0	0	0	0	20
Sequence	0	0	0	0	0	0	0	0	0	0	0	0	24
Alternative	3	0	0	0	0	0	0	0	0	0	0	0	23
Zero or One	0	0	0	3	0	0	0	0	0	0	0	0	24
One or More	10	0	0	8	0	0	0	0	0	0	0	0	34
Zero or More	0	0	0	7	0	0	0	0	0	0	0	0	38
Negated	0	0	0	0	0	0	0	0	0	0	0	0	73
Total	13	0	0	18	0	0	0	0	0	0	0	0	236

Table 6.5: Compliance test results with BeSEPPI

Next, we take a more detailed look at the problems that Virtuoso is currently facing with property path expressions. As already observed by Skubella et al. (2019), Virtuoso produces errors for zero-or-one, zero-or-more, and one-or-more property paths that contain two variables. Furthermore, the error messages state that the transitive start is not given. Therefore, we come to the same conclusion as Skubella et al. (2019) that these features were most likely left out on purpose since Virtuoso is based on relational databases, and it would require huge joins to answer such queries. Moreover, we have noticed that the errors for inverse negated property paths (reported in (Skubella et al., 2019)) have been fixed in the current OpenLink Virtuoso release.

Virtuoso produces 10 incomplete results when evaluating one-or-more property path queries. As already discovered by Skubella et al. (2019), they all cover cases with cycles and miss the start node of the property path, indicating that the one-or-more property path might be implemented by evaluating the zero-or-more property path first and simply removing the start node from the computed result. Finally, in contrast to the results of Skubella et al. (2019), we have found that the current version of OpenLink Virtuoso generates wrong answers for queries that contain alternative property paths. Virtuoso generates incomplete results for three alternative property path queries, which differ from the results of Fuseki and SparqLog by missing all duplicates that should have been generated.

To conclude, while SparqLog and Fuseki handle all considered queries from the three chosen benchmarks correctly, Virtuoso produces a significant number of errors.

6.6.3 Query Answering Performance

Experimental Setup. Our benchmarks were executed on a system running openSUSE Leap 15.2 with dual Intel(R) Xeon(R) Silver 4314 16-core CPUs, clocked at 3.4 GHz, with 512GB RAM, of which 256GB were reserved for the system under test, and 256GB for the operating system. For each system, we set a time-out of 900s. We start each benchmark by repeating the same warm-up queries 5 times and by 5 times loading and deleting the graph instance. Furthermore, we did 5 repetitions of each query (each time deleting and reloading the dataset). We use Apache Jena Fuseki 3.15.0, Virtuoso Open Source Edition 7.2.5, and Stardog 7.7.1 for our experiments. Vadalog loads and queries the databases simultaneously. Hence, to perform a fair comparison with competing systems, we compare their total loading and querying time to the total time that SparqLog needs to answer the query. Since loading includes index building and many more activities, we delete and reload the database each time we run a query (independent of warm-up or benchmark queries).

Performance on General SPARQL Queries

SP2Bench is a benchmark targeting query optimization and computation-intensive queries. We have visualized the result in Figure 6.5, finding that SparqLog reaches competitive performance with Virtuoso and significantly outperforms Fuseki on most queries.

Performance on Property Path Queries

Since current SPARQL benchmarks provide only rudimentary coverage of property path expressions, we have evaluated SparqLog, Fuseki, and Virtuoso using the gMark benchmark generator (Bagan et al., 2017), a domain- and language-independent graph instance and query workload generator which focuses explicitly on path queries, i.e., queries over property paths. We have evaluated SparqLog's, Fuseki's, and Virtuoso's path query performance on the *test*[7] and *social*[8] demo scenarios. Each of these two demo scenarios provides 50 SPARQL queries and a graph instance. Since the graph instances consist of triples of entity and relation IDs, we had to translate the graph instance to RDF by replacing any entity ID α with *<http://ex.org/gMark/*α*>* and any relation ID β with *<http://ex.org/gMark/p*β*>*. Table 6.6 provides further details on the benchmarks we used to evaluate a system's query execution time.

[7]`https://github.com/gbagan/gMark/tree/master/demo/test` (last visited 09/25/2023)
[8]`https://github.com/gbagan/gMark/tree/master/demo/social` (last visited 09/25/2023)

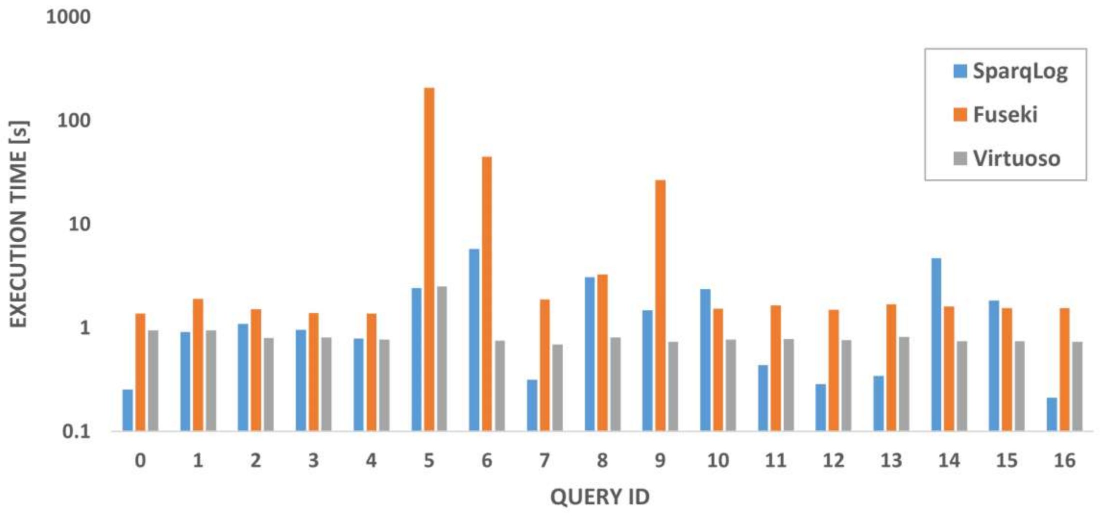

Figure 6.5: SP2Bench benchmark results

Benchmark	#Triples	#Predicates	#Queries
Social (gMark)	226,014	27	50
Test (gMark)	78,582	4	50
SP2Bench	50,168	57	17

Table 6.6: Benchmark statistics

System	SparqLog	Fuseki	Virtuoso
#Not Supported	0	0	12
#Time- and Mem-Outs	1	16	1
#Incomplete Results	0	0	16
Total	1	16	29

Table 6.7: Benchmark results on Social (gMark)

System	SparqLog	Fuseki	Virtuoso
#Not Supported	0	0	9
#Time- and Mem-Outs	1	21	5
#Incomplete Results	0	0	4
Total	1	21	18

Table 6.8: Benchmark results on Test (gMark)

Table 6.7 and 6.8 reveal the results on the gMark benchmarks. Specifically, the tables state the number of queries of the respective benchmark that a system (1) does not support,

(2) answered with a time- or mem-out (out-of-memory) exception, or (3) answered with an incomplete result. Furthermore, the tables present the total number of queries that could not be (correctly) answered by the systems. Figures 6.6 and 6.7 visualize the query execution time of the three systems per benchmarks. A bar reaching 900s represents a time-out. A missing bar represents a mem-out, a faulty result, or that a query was not supported. We have excluded Query 31 from the gMark social benchmark and Query 15 from the gMark test benchmark, as none of the three systems managed to answer these queries. In the following, we compare the results of the three systems on gMark:

Virtuoso could not (correctly) answer 48 of the in total 100 queries of the gMark Social and Test benchmark. Thus, it could not correctly answer almost half of the queries provided by both gMark benchmarks, which empirically reveals its dramatic limitations in answering complex property path queries. In 20 of these 48 cases, Virtuoso returned an incomplete result. While in solely 3 incomplete result cases, Virtuoso missed solely one tuple in the returned result multi-set; in the remaining 17 incomplete result cases, Virtuoso produces either the result tuple *null* or an empty result multi-set instead of the correct non-null/non-empty result multi-set. In the other 28 cases, Virtuoso failed either due to a time-, mem-out, or due to not supporting a property path with two variables. This exemplifies severe problems with handling property path queries.

Fuseki suffered on 37 of the in total 100 queries of the gMark Social and Test benchmark a time-out (i.e., took longer than 900s for answering the queries). Thus, it timed out on more than a third of gMark queries, which empirically reveals its significant limitations in answering complex property path queries.

SparqLog managed to answer 98 of gMark's (in total 100) queries within less than 200s and timed out on solely 2 queries (see Figures 6.6 and 6.7). These results reveal our system's strong ability to answer queries containing complex property paths. Furthermore, each time when both Fuseki and SparqLog returned a result, the results were equal, even further empirically confirming the correctness of our system (i.e., that our system follows the SPARQL standard).

In conclusion, the benchmark results on SP2Bench, GMark (Social), and GMark (Test) show that SparqLog (1) is highly competitive with Virtuoso on regular queries with respect to query execution time (see Figure 6.5), (2) follows the SPARQL standard much more accurately than Virtuoso and supports more property path queries than Virtuoso (see Tables 6.7 and 6.8), and (3) dramatically outperforms Fuseki on query execution while keeping its ability to follow the SPARQL standard accurately.

6.6.4 Ontological Reasoning Performance

One of the main advantages of our SparqLog system is that it provides a uniform and consistent framework for reasoning and querying Knowledge Graphs. We, therefore, wanted to measure the performance of query answering in the presence of an ontology. Since Fuseki and Virtuoso do not provide such support, we compare SparqLog with

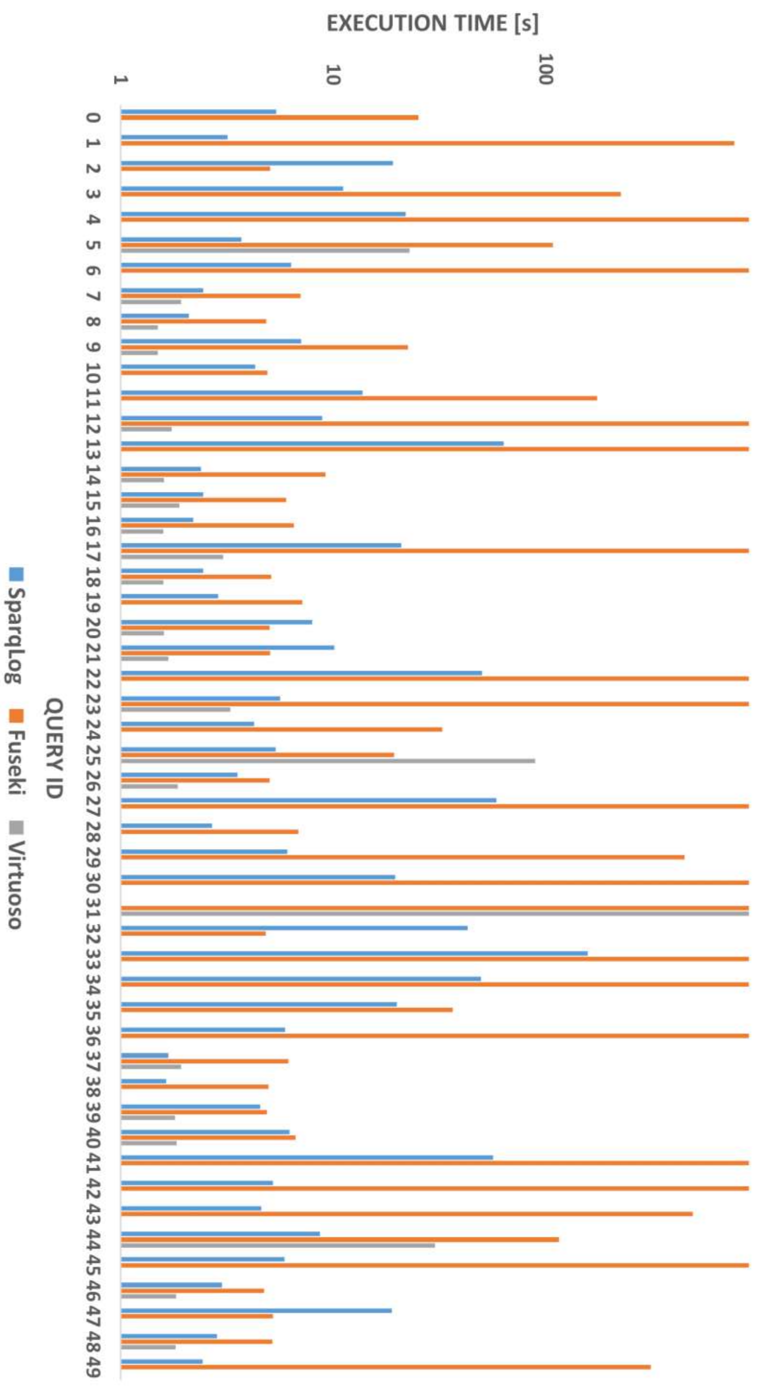

Figure 6.6: gMark Social benchmark results

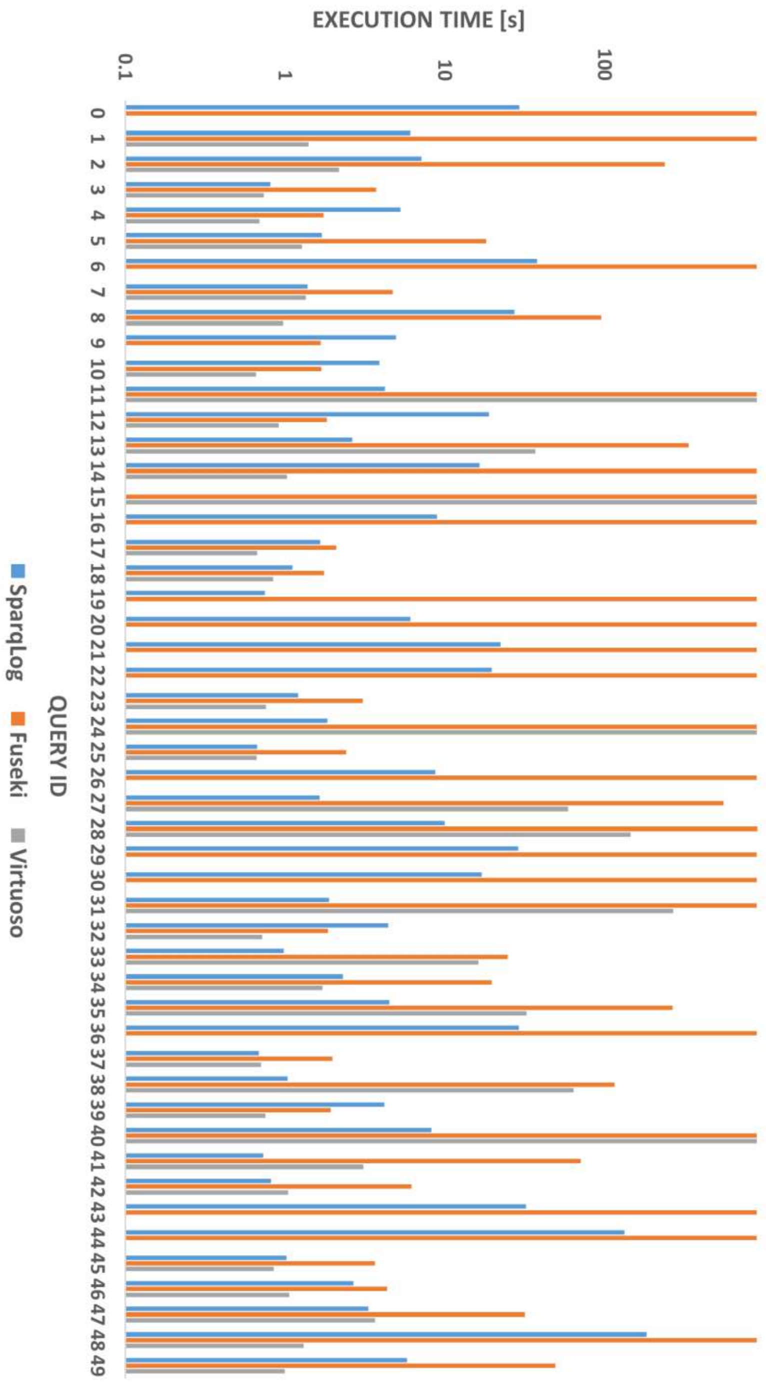

Figure 6.7: gMark Test benchmark results

Stardog, a commonly accepted state-of-the-art system for reasoning and querying within the Semantic Web. Furthermore, we created a new ontology benchmark for reasoning and querying KGs by extending the SP2Bench dataset with (*i*) ontological concepts such as subPropertyOf and subClassOf, and (*ii*) ten queries containing these ontological concepts together with property paths. We list the used ontologies and queries below:

```
# Ontology
bench:Journal rdfs:subClassOf foaf:Document.
bench:Proceedings rdfs:subClassOf foaf:Document.
bench:Inproceedings rdfs:subClassOf foaf:Document.
bench:Article rdfs:subClassOf foaf:Document.
bench:Www rdfs:subClassOf foaf:Document.
bench:MastersThesis rdfs:subClassOf foaf:Document.
bench:PhDThesis rdfs:subClassOf foaf:Document.
bench:Incollection rdfs:subClassOf foaf:Document.
bench:Book rdfs:subClassOf foaf:Document.
dc:title rdfs:subPropertyOf dc:description.
swrc:volume rdfs:subPropertyOf dc:description.
dcterms:issued rdfs:subPropertyOf dc:description.

% Query 0
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT *
WHERE {?s a foaf:Document.}

% Query 1
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT *
WHERE {?s dc:description ?o.}

% Query 2
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX swrc: <http://swrc.ontoware.org/ontology#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT distinct *
WHERE {
        foaf:Document
        ^rdf:type/swrc:journal/dc:description
        ?o.
}

% Query 3
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX swrc: <http://swrc.ontoware.org/ontology#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
```

```
SELECT *
WHERE {
        foaf:Document
        ^rdf:type/swrc:journal/dc:description
        ?o.
}

% Query 4
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT distinct * WHERE {
        ?s
        (^dc:creator/dc:creator)+
        <http://localhost/persons/Paul_Erdoes>.
}

% Query 5
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX swrc: <http://swrc.ontoware.org/ontology#>
SELECT distinct * WHERE {
        ?s
        (^dc:creator/swrc:journal/^swrc:journal/dc:creator)+
        <http://localhost/persons/Paul_Erdoes>.
}

% Query 6
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT distinct * WHERE {
        <http://localhost/persons/Paul_Erdoes>
        (^dc:creator/dc:creator)+
        ?o.
}

% Query 7
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX swrc: <http://swrc.ontoware.org/ontology#>
SELECT distinct * WHERE {
        <http://localhost/persons/Paul_Erdoes>
        (^dc:creator/swrc:journal/^swrc:journal/dc:creator)+
        ?o.
 }

% Query 8
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT distinct * WHERE {
        ?s (^dc:creator/dc:creator)+ ?o.
 }

% Query 9
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX swrc: <http://swrc.ontoware.org/ontology#>
SELECT distinct * WHERE {
        ?s
        (^dc:creator/swrc:journal/^swrc:journal/dc:creator)+
        ?o.
}
```

Furthermore, observe that SP2Bench's dataset represents people as blank nodes (such as `_:Paul_Erdoes`). However, comparing query results containing blank nodes is challenging, as this basically comes down to finding a mapping between the internal system-specific blank node names, which may require exponential time (see Section 6.6.2 for details). Thus, to simplify the query result comparisons, we exchange blank nodes `_:BName` with the name `BName` by IRIs of the form `<http://localhost/persons/BName>` in our ontology benchmark dataset. For example, we use `<http://localhost/persons/Paul_Erdoes>` instead of the blank node `_:Paul_Erdoes`.

Figure 6.8 shows the outcome of our ontology benchmark. In summary, we note that SparqLog is faster than Stardog on most queries. Particularly interesting are Queries 4 and 5, which contain recursive property path queries with two variables. Our engine needs on Query 4 only about a fifth of the execution time of Stardog, and it can even answer Query 5, on which Stardog times outs (using a timeout of 900s). On the other queries, Stardog and SparqLog perform similarly.

To conclude, our new SparqLog system not only follows the SPARQL standard but also shows good performance. Even though SparqLog is a full-fledged, general-purpose Knowledge Graph management system and neither a specialized SPARQL engine nor a specialized ontological reasoner, it is highly competitive with state-of-the-art SPARQL engines and reasoners, and it even outperforms them in answering property path queries and particularly hard cases.

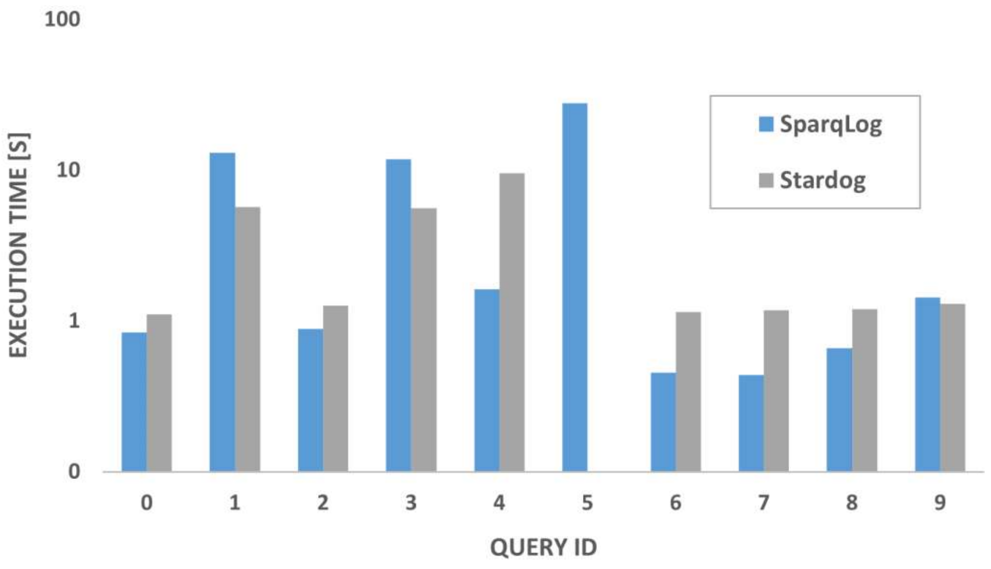

Figure 6.8: Ontology benchmark results

6.7 Summary

In this work we have reached our third and final research goal (G3), by developing SparqLog, a uniform and consistent KG management system that satisfies the requirements (R1–R5) of the SW and DB community. To satisfy R1 and R2, we have (R1) provided a uniform and fairly complete theoretical translation of SPARQL into Warded Datalog$^{\pm}$ that (R2) follows the SPARQL standard by considering bag semantics. Additionally, we have proved the correctness of this translation. Furthermore, as Warded Datalog$^{\pm}$ supports existential quantification, choosing it as the base language allows for ontological reasoning under the OWL 2 QL profile, as required by R3. Moreover, as Warded Datalog$^{\pm}$ is a Datalog dialect, it trivially satisfies R4 by allowing for full recursion. In addition, as Warded Datalog$^{\pm}$ provides an implemented system, namely Vadalog, we could introduce the SparqLog engine, an implementation of our SPARQL translation on top of Vadalog, thereby satisfying R5. Finally, we have provided an extensive experimental evaluation, considering the compliance of our system to the SPARQL standard and its query execution time (also under ontologies). We note that the contribution of the SparqLog engine can be seen in two ways: (1) as a stand-alone translation engine for SPARQL into Warded Datalog$^{\pm}$, and (2) as a full Knowledge Graph engine by using our translation engine together with the Vadalog system. To facilitate the comfortable reuse of our work, we provide SparqLog's source code in a public GitHub repository[9]. Thereby, we have bridged the data management divide, bringing the KG management systems of the DB and SW communities closer together.

[9]https://github.com/joint-kg-labs/SparqLog

CHAPTER 7

Conclusion and Future Work

In this dissertation, we have identified three dimensions dividing KG research from the ML, DB, and SW communities, namely the reasoning, scalability, and data management divides (see Chapter 1). To overcome these divisions, we formulated desired properties of potential solutions by deriving three research goals G1–G3 (see Chapter 1). The key aim of this work is to break down barriers across communities and their research on KGs. Thus, we briefly recall the identified divisions and derived research goals before summarizing our proposed solutions in Section 7.1. Next, we study the broader impact of our achieved goals and point to promising future research directions in Section 7.2. Finally, we discuss concrete open research problems together with future work in Section 7.3.

7.1 Conclusion

G1: Solving the Reasoning Divide. The reasoning divide states that while the DB and SW communities employ a broad set of logical rules to express the key properties of stored data, contemporary KGEs from the ML community are severely constrained in their expressivity of such rules (cf., Table 1.1). Due to this restriction, (G1) more expressive KGEs that are capable of capturing all core inference rules (which are defined in Section 2.1) are required. In order to overcome the reasoning divide, Chapter 4 introduces the spatio-functional embedding model ExpressivE that captures rules via spatial relations of hyper-parallelograms. Thereby, ExpressivE embeddings offer an intuitive and consistent geometric interpretation of their captured rules. In Theorem 4.2.2, Theorem 4.2.4, and Corollary 4.2.5, we exploit the offered geometric interpretation to prove that ExpressivE captures all core inference rules. Thereby, we do not only reach G1 but find the first KGE capable of capturing both general composition and hierarchy jointly (cf., Table 1.1). In addition, we prove that our model is fully expressive, i.e., it can represent any graph, making ExpressivE the first KGE with this capability that supports composition. Finally, we evaluate ExpressivE on the two standard KGC benchmarks and find that ExpressivE is competitive with state-of-the-art gKGEs on FB15k-237 and even significantly outperforms them on WN18RR. Chapter 4 provides more details on each of these results and additional theoretical and empirical analyzes.

G2: Solving the Scalability Divide. The scalability divide poses another critical barrier between KG research efforts, as it sheds light on the efficiency problems of current KGEs, hindering their application on enormous KGs offered by the SW and DB communities. These efficiency problems arise from relying on (*i*) more complex embedding spaces or (*ii*) high embedding dimensionalities for strong prediction performance, raising the space and time requirements of the respective KGEs (see Section 3.2). Thus, the scalability problem calls for (G2) the design of highly resource-efficient models. Based on ExpressivE, Chapter 5 introduces the lightweight SpeedE model that addresses both efficiency problems simultaneously. SpeedE is a *Euclidean* gKGE (addressing Point A) that removes redundant parameters from ExpressivE, halving its inference time while still capturing all core inference rules (proven in Theorem 5.1.1). To address Point B, SpeedE's distance function introduces scalar parameters that allow for more *flexible* representations, raising its KGC performance using low-dimensional embeddings while keeping ExpressivE's intuitive geometric interpretation. We evaluate SpeedE on the three standard KGC benchmarks, finding that it is competitive with state-of-the-art gKGEs on FB15k-237 and even outperforms them significantly on WN18RR and the massive YAGO3-10 benchmark. Most importantly, we find that SpeedE preserves ExpressivE's KGC performance on WN18RR with solely a fourth of the parameters and a fifth of the training time of ExpressivE (Table 5.11, also cf. Section 5.3.3). In total, we reach G2 by proposing the highly scalable SpeedE model that maintains low space and time requirements while reaching strong KGC performance using low-dimensional embeddings. Chapter 5 presents an elaborate discussion on each of these results together with additional findings.

G3: Solving the Data Management Divide. The data management divide reveals that the modeling, querying, and reasoning frameworks for KGs in SW and DB systems are incompatible. Therefore, combining KGs can be a difficult task, demanding (G3) a uniform and consistent KG management system that satisfies the requirements (R1–R5) of both communities (introduced in Section 1.3). Chapter 6 tackles the data management divide by proposing the SparqLog system, which translates the standard modeling (RDF) and querying (SPARQL) languages of the SW into a DB language (Warded Datalog$^{\pm}$). Thereby, SparqLog satisfies the first SW requirement (R1), namely the support of the most commonly used SPARQL features. For a list of its covered SPARQL features, see Table 6.1. In addition, we prove and empirically validate that SparqLog complies with the SPARQL standard. Thus, our system meets the second SW requirement (R2), namely the support of bag semantics as specified in the standard. Furthermore, Warded Datalog$^{\pm}$ straight-forwardly allows the expression of SW ontologies (under OWL 2 QL). Thus, SparqLog accomplishes the last SW requirement, i.e., the support of ontological reasoning. As full recursion is the main feature of Datalog, the DB requirement to support full recursion (R4) is trivially satisfied as Warded Datalog$^{\pm}$ is a

Datalog dialect. Finally, to comply with the last requirement (R5), we have implemented the SparqLog system to allow for its usage in real-world applications. Additionally, we have benchmarked SparqLog's query-execution time, finding that it is comparable with other SPARQL systems, such as Fuseki or Virtuoso, and reasoning systems, such as Stardog, outperforming them significantly on complex queries containing recursive property paths and involving ontologies. Ultimately, we find that SparqLog satisfies all identified requirements (R1–R5) and, thereby, reaches G3. Chapter 6 provides full details on the briefly presented results.

Final Discussion. In total, we have identified and studied three open problems that divide the KG research between the ML, DB, and SW communities. Toward bridging these divisions, we expressed favorable properties of potential solutions in the form of research goals (G1–G3). We reached G1–G3 by proposing concepts for a KG management system and KGEs that we (*i*) studied theoretically to reveal their strong representation and inference capabilities; (*ii*) implemented into practical systems that we made publicly available to allow for their comfortable reuse[1] [2] [3]; and (*iii*) benchmarked to show their superiority over state-of-the-art solutions, in terms of prediction performance and efficiency. Thus, by proposing solutions to overcome the reasoning, scalability, and data management divide, our work contributes toward breaking down the walls separating different communities and extending the foundations for inter-disciplinary KG research while solving vital open problems of the real world.

7.2 Broader Picture and Future Directions

This section switches from the view of individual research results to a broader, holistic view of this dissertation's results and discusses promising future directions. Fundamentally, any KG management systems should allow for at least the following crucial capabilities, including the ability to (*i*) *represent* knowledge about the real world, (*ii*) capture and *reason* over data properties, and (*iii*) *query* its stored information. Although these three parts are at the core of any data management research, many complex challenges remain.

Querying and Reasoning in Incomplete KGs. First, as KGs are inherently incomplete, solutions for querying incomplete KGs need to be found. Although KGEs have been developed to answer simple queries over incomplete KGs, they lack the capabilities to capture and reason over vital data properties, such as general composition rules (cf., Table 1.1). We identified this challenge as the reasoning divide and bridged it by proposing the spatio-functional embedding model ExpressivE (see Chapter 4 for details) that, among its favorable representation capabilities, can represent any graph (i.e., is fully expressive) and can capture relevant rules for modeling key data properties (i.e., captures the core inference rules).

[1]ExpressivE: https://github.com/AleksVap/ExpressivE
[2]SpeedE: https://github.com/AleksVap/SpeedE
[3]SparqLog: https://github.com/joint-kg-labs/SparqLog

Complex Query Answering. Still more challenges remain, particularly most KGEs (Trouillon et al., 2016; Sun et al., 2019; Zhang et al., 2019; Lu and Hu, 2020; Abboud et al., 2020; Pavlović and Sallinger, 2023b, 2024c; Charpenay and Schockaert, 2024) solely focus on a relatively restricted form of query answering, namely the problem of KGC that approximately corresponds to answering atomic queries over binary relations in the DB world; and queries solely containing a single triple pattern in the SW world (see Section 2.1 for a full discussion). Of course, in practice, more complex queries are required. Thus, a recent line of works studied generalizations of KGC by proposing embedding-based approaches for answering more complex queries. Approaches for complex query answering over incomplete KGs include GQE (Hamilton et al., 2018), which supports admissible conjunctive queries; Query2Box (Ren et al., 2020) and CQD (Arakelyan et al., 2021), which additionally allow disjunction in queries; and BetaE (Ren and Leskovec, 2020), ConE (Zhang et al., 2021), MLP (Amayuelas et al., 2022), and Var2Vec (Wang et al., 2023), which allow negation and subsequently reduce the training and inference times.

Efficient Answering of Complex Queries. Still, the efficiency of most embedding-based approaches for complex query answering over incomplete KGs is limited as they require either (*i*) millions of training queries to achieve state-of-the-art performance (Zhang et al., 2021; Amayuelas et al., 2022) or (*ii*) expensively pre-trained gKGEs as a foundation to their approach (Arakelyan et al., 2021; Wang et al., 2023). Even more, we already identified dramatic efficiency problems in the simpler KGC setting, as contemporary gKGEs rely either on (*i*) more complex embedding spaces or (*ii*) high embedding dimensionalities to reach state-of-the-art KGC performance (see Section 3.2). We have identified this fundamental challenge as the scalability divide and bridged this gap by proposing our lightweight SpeedE model (see Chapter 5 for details) that addresses both efficiency problems simultaneously, maintaining low space and time requirements while reaching strong KGC performance using low-dimensional embeddings. Our work on efficient and scalable gKGEs (see Chapter 5) directly benefits many approaches for answering complex queries (Arakelyan et al., 2021; Wang et al., 2023), including the current state of the art, Var2Vec (Wang et al., 2023), that trains additional parameters on top of pre-trained gKGEs, extending them to answer first-order logic queries that use conjunction, disjunction, existential quantification, and negation.

Managing KGs in the Wild. In the real world, representation, reasoning, and querying frameworks for KGs are primarily developed by the DB and the SW communities. However, these frameworks are already incompatible with regard to their used representation formalisms, querying languages, and reasoning capabilities. We identified this problem as the data management divide and solved it by proposing the SparqLog system (see Chapter 6 for details) that translates the standard modeling (RDF) and querying (SPARQL) languages of the SW into a DB language (Warded Datalog$^{\pm}$). Thereby, our system allows to combine the management of KGs from different communities while allowing to query them using SPARQL, the standard querying language of the SW, and simultaneously reason over rules defined in Warded Datalog$^{\pm}$. In contrast to other systems of the SW, e.g., Fuseki, Virtuoso, and Stardog, defining ontologies with such

Warded Datalog$^{\pm}$ rules allows SparqLog to exploit the complete power (e.g., full recursion) of DB frameworks to capture and reason over data properties, even when querying KGs with SPARQL. In addition, although our system is not a full-fledged SPARQL system but a uniform and consistent system for querying and reasoning over KGs, it reaches highly competitive performance compared to state-of-the-art SPARQL systems. Thus, we find that SparqLog does not solely bridge the mismatch between DB- and SW-based KG management frameworks but even combines the advantages of both worlds.

The Big Picture. With this final part in place, looking at the big picture, we see the bridge from SPARQL to Datalog-based engines (stemming from Chapter 6), building upon related work (Polleres and Wallner, 2013; Polleres, 2007; Polleres and Schindlauer, 2007)), hence a reasonably expressive fragment of SW queries. Adding to that is the well-known translation from the wide-spread SW ontology languages of the OWL 2 family to (Warded) Datalog$^{\pm}$ (Arenas et al., 2018), hence a reasonably expressive fragment of SW ontologies. Finally, the initial but growing support of embedding-based reasoning in Datalog$^{\pm}$ engines (Baldazzi et al., 2022), including ExpressivE and SpeedE (Chapters 4 and 5), hence reasonably expressive embedding-based reasoning. Taken together, this allows answering both Datalog-based and SW-based queries (Chapter 6), ontological reasoning in powerful SW languages of the OWL 2 family and Datalog$^{\pm}$ ontologies, and embedding-based reasoning (Chapters 4 and 5) together. Yet, this is not the end of this combination of techniques but rather the beginning, which we will discuss next.

Future Directions. A vital costly data management challenge encountered by many large organizations is how to efficiently query and reason over their collected data, which is typically distributed across various different data sources suited to the needs of specific applications (Bernstein and Haas, 2008). OBDM (Ontology-Based Data Management) mappings (Lenzerini, 2011) are one possible approach to tackle this challenge. Basically, they define a set of rules specifying how to align and integrate diverse data sources under a common ontology (Lenzerini, 2011). Thereby, OBDM mappings make it possible to relate elements of different data sources to ontological concepts in a structured way, allowing for querying and reasoning over heterogeneous data (Lenzerini, 2011). Still, an important challenge of OBDM mappings is how to translate the elements of (often relational) databases to ontological concepts efficiently to facilitate efficient querying and reasoning over abstract ontological representations of the domain rather than application-dependent database systems (Calvanese et al., 2007; Lenzerini, 2011). The work on our SparqLog system naturally aligns with the research on OBDM mappings, as SparqLog establishes a consistent framework for querying and reasoning over both relational databases from the DB and ontologies of the SW side. Thus, a highly interesting path for KG research lies in exploring how to exploit the combined reasoning and querying capabilities of SparqLog, among other Datalog-based KG management systems, to improve the efficiency of OBDM mappings, e.g., by directly exploiting rule-based inference. Another promising direction for the future of KG research lies in the integration of embedding models with KG management systems. In particular, these two frameworks could live in synergy with each other, benefiting from their individual advantages, as we shall discuss next.

Embeddings for KG Management. On the one hand, KG management systems might benefit from the potential of embedding models to (*i*) allow for approximate query answering, especially on very computationally intensive queries; (*ii*) identify semantically similar entities, e.g., for enhancing entity resolution, which is the problem of finding and merging entities that represent the same real-world individual; (*iii*) recommend plausible links to complete the stored KG; and (*iv*) mine rules hidden in the data, enhancing and extending already present ontologies.

KG Management for Embeddings. On the other hand, embedding models might leverage the capabilities of traditional KG management systems, such as their ability to (*i*) answer queries exactly, using these exact answers to improve the embedding model's prediction results; (*ii*) optimize query structures, employing the simplified structures to ease the task of embedding queries and answer them more efficiently; (*iii*) apply logical rules to filter, constrain, or extend prediction results, taking care of, e.g., legal regulations or other constraints of the real world; (*iv*) reason over complex properties, such as recursion or aggregation, which embeddings alone currently have a hard time with; and (*v*) store complex knowledge. In the future, embedding KGs might make all the knowledge stored in the KG management system, including all its triples and logical rules, directly applicable to many downstream tasks. For instance, such knowledge-enriched embeddings might enhance the ones of Large Language Models (LLMs) to mitigate hallucinations, thereby tightening the bond between KG management systems and LLMs.

Embeddings and KG Management. Finally, hybrid systems might very tightly connect KG management systems with embedding models, allowing them to continuously refine each other. For instance, inconsistencies might be detected by a combination of KG management systems and embedding models, triggering updates in the underlying embeddings, triples, and rules of the KG.

Summary. In conclusion, the overall aim of this line of research is to take a step forward at developing systems for efficient and scalable query answering over KGs in various settings, ranging from incomplete graphs to reasoning over entailment regimes. As we have seen, further strengthening the synergy between embeddings and KG management systems will likely enhance the context-awareness and semantic understanding of future solutions for representing, querying, and reasoning over KGs.

7.3 Future Work

Observe that ExpressivE (Pavlović and Sallinger, 2023b) and SpeedE (Pavlović and Sallinger, 2024c), among other gKGEs (Trouillon et al., 2016; Sun et al., 2019; Zhang et al., 2019; Lu and Hu, 2020; Abboud et al., 2020; Charpenay and Schockaert, 2024), focus on the support of individual logical rules, specifically, the core inference rules. However, KG management frameworks of the SW and DB communities, including our SparqLog system, represent key data properties with complex logical languages.

Recently, gKGEs have been developed for specific description logics (Özçep et al., 2020; Mondal et al., 2021; Xiong et al., 2022; Jackermeier et al., 2024), fragments of first-order logic commonly used by the SW community. Yet, less attention was paid to the languages of the DB community, such as Datalog and its dialects. Aside from this, graph neural networks (GNN) — i.e., neural networks designed for graph data — have been very successful on *inductive* KGC (Mai et al., 2021; Teru et al., 2020; Zhu et al., 2021), i.e., KGC where the task is to predict links between entities that have *not* been seen during training. However, GNNs for KGC typically suffer from scalability problems (Pavlović et al., 2024).

In a first step toward efficient graph embeddings that can model data properties with Datalog, we have developed ReshufflE (Pavlović et al., 2024), a GNN for inductive KGC that can learn to exactly predict the triples inferable from an arbitrary set of general composition rules via a bounded number of steps. Based on our results in Pavlović et al. (2024), a highly promising future direction points at the design of graph embeddings for specific Datalog dialects, which would tie KG research of the ML and DB communities even closer together. Even more ambitiously, designing a graph embedding approach for Warded Datalog$^{\pm}$ would be a key achievement for all three considered communities (ML, DB, and SW).

Moreover, observe that SpeedE and ExpressivE use one d-dimensional vector to embed entities and four, respectively, six d-dimensional vectors to embed relations. Thus, ExpressivE and SpeedE have the same space complexity, which is linear in the number of the KG's relations and entities (i.e., $O(d|\boldsymbol{E}| + d|\boldsymbol{R}|)$. A critical limitation of both models is that they use the same dimensionality d for relations and entities. Being able to decouple the relation and entity embedding dimensionalities might be crucial for further raising their efficiency as (i) at an intuitive level, entities are less complex objects than relations (which represent sets of pairs of entities) and therefore (ii) entity embeddings might solely require a lower embedding dimensionality than relation embeddings. Since in real-world KGs, the number of entities is typically much higher than the number of relations, a lower entity dimensionality might further raise the model's efficiency.

Since gKGEs naturally provide a geometric interpretation of their learned rules, how to automatically and efficiently mine these learned rules from the embeddings — to make the implicitly learned knowledge explicit and further raise the model's transparency — remains an open challenge and forms an exciting direction for future work. Another interesting direction points at how to integrate KG embeddings in novel practical applications, such as aligning their learned knowledge with the latent representations of LLMs.

Scaling KGEs and other KGC approaches to massive KG projects, such as DBpedia (Auer et al., 2007) and Wikidata (Vrandecic and Krötzsch, 2014) that currently contain billions of triples[4] [5], is a known open problem in the community. Most evaluation settings of contemporary KGEs consider YAGO3-10 (Mahdisoltani et al., 2015), containing over

[4]https://www.wikidata.org/wiki/Property:P10209 (last visited 31/10/2024)
[5]https://www.dbpedia.org/about/ (last visited 31/10/2024)

a million triples, as their largest KGC benchmark (Abboud et al., 2020; Chami et al., 2020; Charpenay and Schockaert, 2024). While there are many orders of magnitude between the size of YAGO3-10 in comparison to Wikidata and DBpedia, still our results in Chapter 5 show that SpeedE can be effectively trained on YAGO3-10, using rather limited resources. In particular, we solely employ a single GeForce RTX 2080 Ti GPU with 11 GB of RAM. As we saw in Tables 2.1 and 5.10, under the same hyper-parameter setting, the training time of KGEs approximately scales proportionally with the number of triples in the dataset. Specifically, the tables show that SpeedE requires 7s per epoch for WN18RR, containing around 90k triples; 22s for FB15k-237, containing around 270k triples; and 88s for YAGO3-10, containing around 1.1M triples. Thus, in comparison to SpeedE's training time per epoch on WN18RR, SpeedE requires approximately three times longer on FB15k-237 and twelve times longer on YAGO3-10, which contain three and twelve times the triples of WN18RR, respectively. Based on this observation, using stronger resources, such as an NVIDIA A100 with 100 GB of RAM, likely allows SpeedE to train on even larger real-world KGs. Under this setting, SpeedE might be able to embed, for instance, YAGO (Suchanek et al., 2008), a KG containing approximately 15 million triples. In the future, exploring the scalability limits of contemporary KGEs and optimizing their performance further to tackle even larger KGs will be highly interesting.

As to our proposed SparqLog system, we naturally envisage a close to 100% coverage of SPARQL features. Possibly more interesting, we plan to expand on the finding that query plan optimization dramatically affects performance and investigate SPARQL-specific query plan optimization in a unified SPARQL-Datalog$^{\pm}$ system. Finally. as we have observed in Section 6.6.1, no benchmark currently exists that covers all or close to all of the SPARQL 1.1 features. Thus, we note that work on a unified benchmark considering most features would be desirable.

In summary, all investigated fields offer crucial individual problems whose solutions will further strengthen the connection between Knowledge Graph research of the Machine Learning, Database, and Semantic Web communities.

List of Figures

List of Tables

Acronyms

DB	Database.
DLV	Datalog with Disjunction.
gKGE	geometric Knowledge Graph Embedding Model.
IRI	Internationalized Resource Identifier.
KG	Knowledge Graph.
KGC	Knowledge Graph Completion.
KGE	Knowledge Graph Embedding Model.
LLM	Large Language Model.
ML	Machine Learning.
OWL	Web Ontology Language.
OWL 2 QL	OWL 2 Query Logic.
OWL 2 RL	OWL 2 Rules Logic.
RDF	Resource Description Framework.
RDFS	RDF Schema.
SPARQL	SPARQL Protocol and RDF Query Language.
SQL	Structured Query Language.
SW	Semantic Web.
W3C	World Wide Web Consortium.

Bibliography

Abboud, R., Ceylan, İ. İ., Lukasiewicz, T., and Salvatori, T. (2020). BoxE: A box embedding model for knowledge base completion. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.*

Abiteboul, S., Hull, R., and Vianu, V. (1995). *Foundations of Databases.* Addison-Wesley.

Akrami, F., Saeef, M. S., Zhang, Q., Hu, W., and Li, C. (2020). Realistic re-evaluation of knowledge graph completion methods: An experimental study. In Maier, D., Pottinger, R., Doan, A., Tan, W., Alawini, A., and Ngo, H. Q., editors, *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, pages 1995–2010. ACM.

Ali, M., Berrendorf, M., Hoyt, C. T., Vermue, L., Sharifzadeh, S., Tresp, V., and Lehmann, J. (2021). PyKEEN 1.0: A python library for training and evaluating knowledge graph embeddings. *Journal of Machine Learning Research*, 22:82:1–82:6.

Aluç, G., Hartig, O., Özsu, M. T., and Daudjee, K. (2014). Diversified stress testing of RDF data management systems. In Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C. A., Vrandecic, D., Groth, P., Noy, N. F., Janowicz, K., and Goble, C. A., editors, *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I*, volume 8796 of *Lecture Notes in Computer Science*, pages 197–212. Springer.

Amayuelas, A., Zhang, S., Rao, S. X., and Zhang, C. (2022). Neural methods for logical reasoning over knowledge graphs. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Angles, R., Gottlob, G., Pavlović, A., Pichler, R., and Sallinger, E. (2023a). SparqLog: A system for efficient evaluation of SPARQL 1.1 queries via Datalog. *Proc. VLDB Endow.*, 16(13):4240–4253.

Angles, R., Gottlob, G., Pavlović, A., Pichler, R., and Sallinger, E. (2023b). SparqLog: A system for efficient evaluation of SPARQL 1.1 queries via Datalog [experiment, analysis and benchmark]. *CoRR*, abs/2307.06119.

Angles, R. and Gutierrez, C. (2008). The expressive power of SPARQL. In Sheth, A. P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T. W., and Thirunarayan, K., editors, *The Semantic Web - ISWC 2008, 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany, October 26-30, 2008. Proceedings*, volume 5318 of *Lecture Notes in Computer Science*, pages 114–129. Springer.

Angles, R. and Gutierrez, C. (2016a). The multiset semantics of SPARQL patterns. In Groth, P., Simperl, E., Gray, A. J. G., Sabou, M., Krötzsch, M., Lécué, F., Flöck, F., and Gil, Y., editors, *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part I*, volume 9981 of *Lecture Notes in Computer Science*, pages 20–36.

Angles, R. and Gutierrez, C. (2016b). Negation in SPARQL. In Pichler, R. and da Silva, A. S., editors, *Proceedings of the 10th Alberto Mendelzon International Workshop on Foundations of Data Management, Panama City, Panama, May 8-10, 2016*, volume 1644 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Arakelyan, E., Daza, D., Minervini, P., and Cochez, M. (2021). Complex query answering with neural link predictors. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Arenas, M., Gottlob, G., and Pieris, A. (2018). Expressive languages for querying the semantic web. *ACM Trans. Database Syst.*, 43(3):13:1–13:45.

Arenas, M., Gutierrez, C., and Pérez, J. (2009). On the semantics of SPARQL. In Virgilio, R. D., Giunchiglia, F., and Tanca, L., editors, *Semantic Web Information Management - A Model-Based Perspective*, pages 281–307. Springer.

Atzeni, P., Bellomarini, L., Iezzi, M., Sallinger, E., and Vlad, A. (2020). Weaving enterprise knowledge graphs: The case of company ownership graphs. In Bonifati, A., Zhou, Y., Salles, M. A. V., Böhm, A., Olteanu, D., Fletcher, G. H. L., Khan, A., and Yang, B., editors, *Proceedings of the 23rd International Conference on Extending Database Technology, EDBT 2020, Copenhagen, Denmark, March 30 - April 02, 2020*, pages 555–566. OpenProceedings.org.

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. G. (2007). DBpedia: A nucleus for a web of open data. In Aberer, K., Choi, K., Noy, N. F., Allemang, D., Lee, K., Nixon, L. J. B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., and Cudré-Mauroux, P., editors, *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007*, volume 4825 of *Lecture Notes in Computer Science*, pages 722–735. Springer.

Bagan, G., Bonifati, A., Ciucanu, R., Fletcher, G. H. L., Lemay, A., and Advokaat, N. (2017). gMark: Schema-driven generation of graphs and queries. In *33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017*, pages 63–64. IEEE Computer Society.

Baget, J., Leclère, M., Mugnier, M., Rocher, S., and Sipieter, C. (2015). Graal: A toolkit for query answering with existential rules. In Bassiliades, N., Gottlob, G., Sadri, F., Paschke, A., and Roman, D., editors, *Rule Technologies: Foundations, Tools, and Applications - 9th International Symposium, RuleML 2015, Berlin, Germany, August 2-5, 2015, Proceedings*, volume 9202 of *Lecture Notes in Computer Science*, pages 328–344. Springer.

Baget, J., Leclère, M., Mugnier, M., and Salvat, E. (2009). Extending decidable cases for rules with existential variables. In Boutilier, C., editor, *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 677–682.

Baget, J., Leclère, M., Mugnier, M., and Salvat, E. (2011). On rules with existential variables: Walking the decidability line. *Artif. Intell.*, 175(9-10):1620–1654.

Bai, Y., Ying, Z., Ren, H., and Leskovec, J. (2021). Modeling heterogeneous hierarchies with relation-specific hyperbolic cones. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 12316–12327.

Bail, S., Alkiviadous, S., Parsia, B., Workman, D., van Harmelen, M., Gonçalves, R. S., and Garilao, C. (2012). FishMark: A linked data application benchmark. In Fokoue, A., Liebig, T., Goodman, E. L., Weaver, J., Urbani, J., and Mizell, D., editors, *Proceedings of the Joint Workshop on Scalable and High-Performance Semantic Web Systems, Boston, USA, November 11, 2012*, volume 943 of *CEUR Workshop Proceedings*, pages 1–15. CEUR-WS.org.

Balazevic, I., Allen, C., and Hospedales, T. M. (2019a). Multi-relational poincaré graph embeddings. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 4465–4475.

Balazevic, I., Allen, C., and Hospedales, T. M. (2019b). TuckER: Tensor factorization for knowledge graph completion. In Inui, K., Jiang, J., Ng, V., and Wan, X., editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 5184–5193. Association for Computational Linguistics.

Baldazzi, T., Bellomarini, L., Gerschberger, M., Jami, A., Magnanimi, D., Nissl, M., Pavlovic, A., and Sallinger, E. (2022). Vadalog: Overview, extensions and business applications. In Bertossi, L. E. and Xiao, G., editors, *Reasoning Web. Causality, Explanations and Declarative Knowledge - 18th International Summer School 2022,* *Berlin, Germany, September 27-30, 2022, Tutorial Lectures*, volume 13759 of *Lecture Notes in Computer Science*, pages 161–198. Springer.

Bellomarini, L., Sallinger, E., and Gottlob, G. (2018). The Vadalog System: Datalog-based reasoning for knowledge graphs. *Proc. VLDB Endow.*, 11(9):975–987.

Bernstein, P. A. and Haas, L. M. (2008). Information integration in the enterprise. *Commun. ACM*, 51(9):72–79.

Bertossi, L. E., Gottlob, G., and Pichler, R. (2019). Datalog: Bag semantics via set semantics. In Barceló, P. and Calautti, M., editors, *22nd International Conference on Database Theory, ICDT 2019, March 26-28, 2019, Lisbon, Portugal*, volume 127 of *LIPIcs*, pages 16:1–16:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Bizer, C. and Schultz, A. (2009). The berlin SPARQL benchmark. *Int. J. Semantic Web Inf. Syst.*, 5(2):1–24.

Bollacker, K. D., Cook, R. P., and Tufts, P. (2007). Freebase: A shared database of structured general human knowledge. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 1962–1963. AAAI Press.

Bonifati, A., Martens, W., and Timm, T. (2020). An analytical study of large SPARQL query logs. *VLDB J.*, 29(2-3):655–679.

Bordes, A., Usunier, N., García-Durán, A., Weston, J., and Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In Burges, C. J. C., Bottou, L., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 2787–2795.

Brickley, D. and Guha., R. V. (2014). RDF Schema 1.1 (W3C Recommendation). http://www.w3.org/TR/rdf-schema/.

Broscheit, S., Gashteovski, K., Wang, Y., and Gemulla, R. (2020). Can we predict new facts with open knowledge graph embeddings? A benchmark for open link prediction. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J. R., editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 2296–2308. Association for Computational Linguistics.

Calì, A., Gottlob, G., and Kifer, M. (2013). Taming the infinite chase: Query answering under expressive relational constraints. *J. Artif. Intell. Res.*, 48:115–174.

Calì, A., Gottlob, G., and Lukasiewicz, T. (2009). Datalog$^\pm$: A unified approach to ontologies and integrity constraints. In Fagin, R., editor, *Database Theory - ICDT 2009, 12th International Conference, St. Petersburg, Russia, March 23-25, 2009, Proceedings*, volume 361 of *ACM International Conference Proceeding Series*, pages 14–30. ACM.

Calì, A., Gottlob, G., and Pieris, A. (2010a). Advanced processing for ontological queries. *Proc. VLDB Endow.*, 3(1):554–565.

Calì, A., Gottlob, G., and Pieris, A. (2010b). Query answering under non-guarded rules in Datalog$^{\pm}$. In Hitzler, P. and Lukasiewicz, T., editors, *Web Reasoning and Rule Systems - Fourth International Conference, RR 2010, Bressanone/Brixen, Italy, September 22-24, 2010. Proceedings*, volume 6333 of *Lecture Notes in Computer Science*, pages 1–17. Springer.

Calvanese, D., Giacomo, G. D., Lembo, D., Lenzerini, M., and Rosati, R. (2007). Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. Autom. Reason.*, 39(3):385–429.

Cao, Y., Wang, X., He, X., Hu, Z., and Chua, T. (2019). Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In Liu, L., White, R. W., Mantrach, A., Silvestri, F., McAuley, J. J., Baeza-Yates, R., and Zia, L., editors, *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 151–161. ACM.

Cao, Z., Xu, Q., Yang, Z., Cao, X., and Huang, Q. (2021). Dual quaternion knowledge graph embeddings. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 6894–6902. AAAI Press.

Carral, D., Dragoste, I., González, L., Jacobs, C. J. H., Krötzsch, M., and Urbani, J. (2019). VLog: A rule engine for knowledge graphs. In Ghidini, C., Hartig, O., Maleshkova, M., Svátek, V., Cruz, I. F., Hogan, A., Song, J., Lefrançois, M., and Gandon, F., editors, *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II*, volume 11779 of *Lecture Notes in Computer Science*, pages 19–35. Springer.

Chami, I., Wolf, A., Juan, D., Sala, F., Ravi, S., and Ré, C. (2020). Low-dimensional hyperbolic knowledge graph embeddings. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J. R., editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 6901–6914. Association for Computational Linguistics.

Charpenay, V. and Schockaert, S. (2024). Capturing knowledge graphs and rules with octagon embeddings. In Larson, K., editor, *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 3289–3297. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Chen, M. and Zaniolo, C. (2017). Learning multi-faceted knowledge graph embeddings for natural language processing. In Sierra, C., editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 5169–5170. ijcai.org.

Cyganiak, R., Wood, D., and Lanthaler, M. (2014). RDF 1.1 Concepts and Abstract Syntax (W3C Recommendation). https://www.w3.org/TR/rdf11-concepts/.

Demartini, G., Enchev, I., Wylot, M., Gapany, J., and Cudré-Mauroux, P. (2011). BowlognaBench - benchmarking RDF analytics. In Aberer, K., Damiani, E., and Dillon, T. S., editors, *Data-Driven Process Discovery and Analysis - First International Symposium, SIMPDA 2011, Campione d'Italia, Italy, June 29 - July 1, 2011, Revised Selected Papers*, volume 116 of *Lecture Notes in Business Information Processing*, pages 82–102. Springer.

Dettmers, T., Minervini, P., Stenetorp, P., and Riedel, S. (2018). Convolutional 2d knowledge graph embeddings. In McIlraith, S. A. and Weinberger, K. Q., editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 1811–1818. AAAI Press.

Dietz, L., Kotov, A., and Meij, E. (2018). Utilizing knowledge graphs for text-centric information retrieval. In Collins-Thompson, K., Mei, Q., Davison, B. D., Liu, Y., and Yilmaz, E., editors, *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 1387–1390. ACM.

Eiter, T., Fink, M., Krennwallner, T., and Redl, C. (2013). HEX-programs with existential quantification. In Hanus, M. and Rocha, R., editors, *Declarative Programming and Knowledge Management - Declarative Programming Days, KDPD 2013, Unifying INAP, WFLP, and WLP, Kiel, Germany, September 11-13, 2013, Revised Selected Papers*, volume 8439 of *Lecture Notes in Computer Science*, pages 99–117. Springer.

Erling, O., Averbuch, A., Larriba-Pey, J. L., Chafi, H., Gubichev, A., Prat-Pérez, A., Pham, M., and Boncz, P. A. (2015). The LDBC social network benchmark: Interactive workload. In Sellis, T. K., Davidson, S. B., and Ives, Z. G., editors, *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pages 619–630. ACM.

Fagin, R., Kolaitis, P. G., Miller, R. J., and Popa, L. (2005). Data exchange: semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124.

Galárraga, L., Teflioudi, C., Hose, K., and Suchanek, F. M. (2015). Fast rule mining in ontological knowledge bases with AMIE+. *VLDB J.*, 24(6):707–730.

Galárraga, L. A., Teflioudi, C., Hose, K., and Suchanek, F. M. (2013). AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In Schwabe, D., Almeida, V. A. F., Glaser, H., Baeza-Yates, R., and Moon, S. B., editors, *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, pages 413–422. International World Wide Web Conferences Steering Committee / ACM.

Gao, C., Sun, C., Shan, L., Lin, L., and Wang, M. (2020). Rotate3D: Representing relations as rotations in three-dimensional space for knowledge graph embedding. In d'Aquin, M., Dietze, S., Hauff, C., Curry, E., and Cudré-Mauroux, P., editors, *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, pages 385–394. ACM.

Glimm, B. and Ogbuji, C. (2013). SPARQL 1.1 Entailment Regimes (W3C Recommendation). http://www.w3.org/TR/sparql11-entailment/.

Gottlob, G., Orsi, G., Pieris, A., and Simkus, M. (2012). Datalog and its extensions for semantic web databases. In Eiter, T. and Krennwallner, T., editors, *Reasoning Web. Semantic Technologies for Advanced Query Answering - 8th International Summer School 2012, Vienna, Austria, September 3-8, 2012. Proceedings*, volume 7487 of *Lecture Notes in Computer Science*, pages 54–77. Springer.

Gottlob, G. and Pieris, A. (2015). Beyond SPARQL under OWL 2 QL entailment regime: Rules to the rescue. In Yang, Q. and Wooldridge, M. J., editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 2999–3007. AAAI Press.

Green, T. J., Huang, S. S., Loo, B. T., and Zhou, W. (2013). Datalog and recursive query processing. *Found. Trends Databases*, 5(2):105–195.

Hamilton, W. L., Bajaj, P., Zitnik, M., Jurafsky, D., and Leskovec, J. (2018). Embedding logical queries on knowledge graphs. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 2030–2041.

Harris, S. and Seaborne, A. (2013). SPARQL 1.1 Query Language (W3C Recommendation). https://www.w3.org/TR/sparql11-query/.

Hayashi, K. and Shimbo, M. (2017). On the equivalence of holographic and complex embeddings for link prediction. In Barzilay, R. and Kan, M., editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 2: Short Papers*, pages 554–559. Association for Computational Linguistics.

Hayes, P. J. and Patel-Schneider, P. F. (2014). RDF 1.1 Semantics (W3C Recommendation). http://www.w3.org/TR/rdf11-mt/.

Hitchcock, F. L. (1927). The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189.

Jackermeier, M., Chen, J., and Horrocks, I. (2024). Dual box embeddings for the description logic EL^{++}. In Chua, T., Ngo, C., Kumar, R., Lauw, H. W., and Lee, R. K., editors, *Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, May 13-17, 2024*, pages 2250–2258. ACM.

Johnson, D. S. and Klug, A. C. (1984). Testing containment of conjunctive queries under functional and inclusion dependencies. *J. Comput. Syst. Sci.*, 28(1):167–189.

Kazemi, S. M. and Poole, D. (2018). SimplE embedding for link prediction in knowledge graphs. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 4289–4300.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kostylev, E. V., Reutter, J. L., Romero, M., and Vrgoc, D. (2015). SPARQL with property paths. In Arenas, M., Corcho, Ó., Simperl, E., Strohmaier, M., d'Aquin, M., Srinivas, K., Groth, P., Dumontier, M., Heflin, J., Thirunarayan, K., and Staab, S., editors, *The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part I*, volume 9366 of *Lecture Notes in Computer Science*, pages 3–18. Springer.

Lacoste, A., Luccioni, A., Schmidt, V., and Dandres, T. (2019). Quantifying the carbon emissions of machine learning. *CoRR*, abs/1910.09700.

Lacroix, T., Usunier, N., and Obozinski, G. (2018). Canonical tensor decomposition for knowledge base completion. In Dy, J. G. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2869–2878. PMLR.

Lenzerini, M. (2011). Ontology-based data management. In Macdonald, C., Ounis, I., and Ruthven, I., editors, *Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24-28, 2011*, pages 5–6. ACM.

Li, X., Vilnis, L., Zhang, D., Boratko, M., and McCallum, A. (2019). Smoothing the geometry of probabilistic box embeddings. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Lu, H. and Hu, H. (2020). DensE: An enhanced non-abelian group representation for knowledge graph embedding. *CoRR*, abs/2008.04548.

Mahdisoltani, F., Biega, J., and Suchanek, F. M. (2015). YAGO3: A knowledge base from multilingual wikipedias. In *Seventh Biennial Conference on Innovative Data Systems Research, CIDR 2015, Asilomar, CA, USA, January 4-7, 2015, Online Proceedings*. www.cidrdb.org.

Mai, S., Zheng, S., Yang, Y., and Hu, H. (2021). Communicative message passing for inductive relation reasoning. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 4294–4302. AAAI Press.

Miller, G. A. (1995). WordNet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Mondal, S., Bhatia, S., and Mutharaju, R. (2021). EmEL++: Embeddings for EL++ description logic. In Martin, A., Hinkelmann, K., Fill, H., Gerber, A., Lenat, D., Stolle, R., and van Harmelen, F., editors, *Proceedings of the AAAI 2021 Spring Symposium on Combining Machine Learning and Knowledge Engineering (AAAI-MAKE 2021), Stanford University, Palo Alto, California, USA, March 22-24, 2021*, volume 2846 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Morsey, M., Lehmann, J., Auer, S., and Ngomo, A. N. (2011). DBpedia SPARQL benchmark - performance assessment with real queries on real data. In Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N. F., and Blomqvist, E., editors, *The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I*, volume 7031 of *Lecture Notes in Computer Science*, pages 454–469. Springer.

Motik, B., Grau, B. C., Horrocks, I., Wu, Z., Fokoue, A., and Lutz, C. (2012a). OWL 2 Web Ontology Language: Profiles (Second Edition) (W3C Recommendation). http://www.w3.org/TR/owl2-profiles/.

Motik, B., Patel-Schneider, P. F., and Grau, B. C. (2012b). OWL 2 Web Ontology Language: Direct Semantics (Second Edition) (W3C Recommendation). http://www.w3.org/TR/owl2-direct-semantics/.

Motik, B., Patel-Schneider, P. F., and Parsia, B. (2012c). OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax (Second Edition) (W3C Recommendation). http://www.w3.org/TR/owl2-syntax/.

Mumick, I. S., Pirahesh, H., and Ramakrishnan, R. (1990). The magic of duplicates and aggregates. In McLeod, D., Sacks-Davis, R., and Schek, H., editors, *16th International Conference on Very Large Data Bases, August 13-16, 1990, Brisbane, Queensland, Australia, Proceedings*, pages 264–277. Morgan Kaufmann.

Mumick, I. S. and Shmueli, O. (1993). Finiteness properties of database queries. In Orlowska, M. E. and Papazoglou, M. P., editors, *Advances in Database Research - Proceedings of the 4th Australian Database Conference, ADC '93, Griffith University, Brisbane, Queensland, Australia, February 1-2, 1993*, pages 274–288. World Scientific.

Nathani, D., Chauhan, J., Sharma, C., and Kaul, M. (2019). Learning attention-based embeddings for relation prediction in knowledge graphs. In Korhonen, A., Traum, D. R., and Màrquez, L., editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4710–4723. Association for Computational Linguistics.

Nenov, Y., Piro, R., Motik, B., Horrocks, I., Wu, Z., and Banerjee, J. (2015). RDFox: A highly-scalable RDF store. In Arenas, M., Corcho, Ó., Simperl, E., Strohmaier, M., d'Aquin, M., Srinivas, K., Groth, P., Dumontier, M., Heflin, J., Thirunarayan, K., and Staab, S., editors, *The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part II*, volume 9367 of *Lecture Notes in Computer Science*, pages 3–20. Springer.

Nickel, M., Rosasco, L., and Poggio, T. A. (2016). Holographic embeddings of knowledge graphs. In Schuurmans, D. and Wellman, M. P., editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 1955–1961. AAAI Press.

Nickel, M., Tresp, V., and Kriegel, H. (2011). A three-way model for collective learning on multi-relational data. In Getoor, L. and Scheffer, T., editors, *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 809–816. Omnipress.

Özçep, Ö. L., Leemhuis, M., and Wolter, D. (2020). Cone semantics for logics with negation. In Bessiere, C., editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 1820–1826. ijcai.org.

Pavlović, A. (2019). Translating SPARQL to Vadalog. *Bachelor Thesis, Technische Universität Wien.*

Pavlović, A. (2020). Reasoning in knowledge graphs: Bridging databases and the semantic web. In *Diploma Thesis, Technische Universität Wien*. reposiTUm.

Pavlović, A. and Sallinger, E. (2023a). Building bridges: Knowledge graph embeddings respecting logical rules (short paper). In Kimelfeld, B., Martinez, M. V., and Angles, R., editors, *Proceedings of the 15th Alberto Mendelzon International Workshop on Foundations of Data Management (AMW 2023), Santiago de Chile, Chile, May 22-26, 2023*, volume 3409 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Pavlović, A. and Sallinger, E. (2023b). ExpressivE: A spatio-functional embedding for knowledge graph completion. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigal, Rwanda, May 1-5, 2023.*

Pavlović, A. and Sallinger, E. (2024a). Expressive and geometrically interpretable knowledge graph embedding (extended abstract). In *The First Austrian Symposium on AI, Robotics, and Vision (AIROV24).*

Pavlović, A. and Sallinger, E. (2024b). Raising the efficiency of knowledge graph embeddings while respecting logical rules (short paper). In Montoya, G., Sallinger, E., and Vargas-Solar, G., editors, *Proceedings of the 16th Alberto Mendelzon International Workshop on Foundations of Data Management (AMW 2024), Mexico City, Mexico, September 30 - October 4, 2024*, CEUR Workshop Proceedings. CEUR-WS.org.

Pavlović, A. and Sallinger, E. (2024c). SpeedE: Euclidean geometric knowledge graph embedding strikes back. In Duh, K., Gomez, H., and Bethard, S., editors, *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 69–92, Mexico City, Mexico. Association for Computational Linguistics.

Pavlović, A., Sallinger, E., and Schockaert, S. (2024). Differentiable reasoning about knowledge graphs with region-based graph neural networks. *CoRR*, abs/2406.09529.

Polleres, A. (2007). From SPARQL to rules (and back). In Williamson, C. L., Zurko, M. E., Patel-Schneider, P. F., and Shenoy, P. J., editors, *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 787–796. ACM.

Polleres, A. and Schindlauer, R. (2007). DLVHEX-SPARQL: A SPARQL compliant query engine based on DLVHEX. In Polleres, A., Pearce, D., Heymans, S., and Ruckhaus, E., editors, *Proceedings of the ICLP'07 Workshop on Applications of Logic Programming to the Web, Semantic Web and Semantic Web Services, ALPSWS 2007, Porto, Portugal, September 13th, 2007*, volume 287 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Polleres, A. and Wallner, J. P. (2013). On the relation between SPARQL1.1 and answer set programming. *J. Appl. Non Class. Logics*, 23(1-2):159–212.

Prud'hommeaux, E. and Seaborne, A. (2008). SPARQL Query Language for RDF (W3C Recommendation). https://www.w3.org/TR/rdf-sparql-query/.

Przymus, P., Boniewicz, A., Burzanska, M., and Stencel, K. (2010). Recursive query facilities in relational databases: A survey. In Zhang, Y., Cuzzocrea, A., Ma, J., Chung, K., Arslan, T., and Song, X., editors, *Database Theory and Application, Bio-Science and Bio-Technology - International Conferences, DTA and BSBT 2010, Held as Part of the Future Generation Information Technology Conference, FGIT 2010, Jeju Island, Korea, December 13-15, 2010. Proceedings*, volume 118 of *Communications in Computer and Information Science*, pages 89–99. Springer.

Ren, H., Hu, W., and Leskovec, J. (2020). Query2box: Reasoning over knowledge graphs in vector space using box embeddings. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Ren, H. and Leskovec, J. (2020). Beta embeddings for multi-hop logical reasoning in knowledge graphs. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems 33: Annual Conference* *on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.*

Ruffinelli, D., Broscheit, S., and Gemulla, R. (2020). You CAN teach an old dog new tricks! On training knowledge graph embeddings. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Saleem, M., Mehmood, Q., and Ngomo, A. N. (2015). FEASIBLE: A feature-based SPARQL benchmark generation framework. In Arenas, M., Corcho, Ó., Simperl, E., Strohmaier, M., d'Aquin, M., Srinivas, K., Groth, P., Dumontier, M., Heflin, J., Thirunarayan, K., and Staab, S., editors, *The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part I*, volume 9366 of *Lecture Notes in Computer Science*, pages 52–69. Springer.

Saleem, M., Szárnyas, G., Conrads, F., Bukhari, S. A. C., Mehmood, Q., and Ngomo, A. N. (2019). How representative is a SPARQL benchmark? An analysis of RDF triplestore benchmarks. In Liu, L., White, R. W., Mantrach, A., Silvestri, F., McAuley, J. J., Baeza-Yates, R., and Zia, L., editors, *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 1623–1633. ACM.

Schmidt, M., Hornung, T., Lausen, G., and Pinkel, C. (2008). SP2Bench: A SPARQL performance benchmark. *CoRR*, abs/0806.4627.

Schneider, M. (2012). OWL 2 Web Ontology Language: RDF-Based Semantics (Second Edition) (W3C Recommendation). http://www.w3.org/TR/owl2-rdf-based-semantics/.

Shen, T., Zhang, F., and Cheng, J. (2022). A comprehensive overview of knowledge graph completion. *Knowl. Based Syst.*, 255:109597.

Skubella, A., Janke, D., and Staab, S. (2019). BeSEPPI: Semantic-based benchmarking of property path implementations. In Hitzler, P., Fernández, M., Janowicz, K., Zaveri, A., Gray, A. J. G., López, V., Haller, A., and Hammar, K., editors, *The Semantic Web - 16th International Conference, ESWC 2019, Portorož, Slovenia, June 2-6, 2019, Proceedings*, volume 11503 of *Lecture Notes in Computer Science*, pages 475–490. Springer.

Socher, R., Chen, D., Manning, C. D., and Ng, A. Y. (2013). Reasoning with neural tensor networks for knowledge base completion. In Burges, C. J. C., Bottou, L., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 926–934.

Song, T., Luo, J., and Huang, L. (2021). Rot-pro: Modeling transitivity by projection in knowledge graph embedding. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 24695–24706.

Subramanian, S. and Chakrabarti, S. (2018). New embedded representations and evaluation protocols for inferring transitive relations. In Collins-Thompson, K., Mei, Q., Davison, B. D., Liu, Y., and Yilmaz, E., editors, *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 1037–1040. ACM.

Suchanek, F. M., Kasneci, G., and Weikum, G. (2008). YAGO: A large ontology from wikipedia and wordnet. *J. Web Semant.*, 6(3):203–217.

Sun, Z., Deng, Z., Nie, J., and Tang, J. (2019). RotatE: Knowledge graph embedding by relational rotation in complex space. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Szárnyas, G., Izsó, B., Ráth, I., and Varró, D. (2018a). The train benchmark: cross-technology performance evaluation of continuous model queries. *Softw. Syst. Model.*, 17(4):1365–1393.

Szárnyas, G., Prat-Pérez, A., Averbuch, A., Marton, J., Paradies, M., Kaufmann, M., Erling, O., Boncz, P. A., Haprian, V., and Antal, J. B. (2018b). An early look at the LDBC social network benchmark's business intelligence workload. In Arora, A., Bhattacharya, A., Fletcher, G. H. L., Larriba-Pey, J. L., Roy, S., and West, R., editors, *Proceedings of the 1st ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA), Houston, TX, USA, June 10, 2018*, pages 9:1–9:11. ACM.

Teru, K. K., Denis, E. G., and Hamilton, W. L. (2020). Inductive relation prediction by subgraph reasoning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 9448–9457. PMLR.

Toutanova, K. and Chen, D. (2015). Observed versus latent features for knowledge base and text inference. In Allauzen, A., Grefenstette, E., Hermann, K. M., Larochelle, H., and Yih, S. W., editors, *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality, CVSC 2015, Beijing, China, July 26-31, 2015*, pages 57–66. Association for Computational Linguistics.

Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., and Bouchard, G. (2016). Complex embeddings for simple link prediction. In Balcan, M. and Weinberger, K. Q., editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2071–2080. JMLR.org.

Tucker, L. R. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311.

Vianu, V. (2021). Datalog unchained. In Libkin, L., Pichler, R., and Guagliardo, P., editors, *PODS'21: Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Virtual Event, China, June 20-25, 2021*, pages 57–69. ACM.

Vilnis, L., Li, X., Murty, S., and McCallum, A. (2018). Probabilistic embedding of knowledge graphs with box lattice measures. In Gurevych, I. and Miyao, Y., editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 263–272. Association for Computational Linguistics.

Vrandecic, D. and Krötzsch, M. (2014). Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Wang, D., Chen, Y., and Cuenca Grau, B. (2023). Efficient embeddings of logical variables for query answering over incomplete knowledge graphs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(4):4652–4659.

Wang, K., Liu, Y., Lin, D., and Sheng, M. (2021). Hyperbolic geometry is not necessary: Lightweight Euclidean-based models for low-dimensional knowledge graph embeddings. In Moens, M., Huang, X., Specia, L., and Yih, S. W., editors, *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, pages 464–474. Association for Computational Linguistics.

Wang, Q., Mao, Z., Wang, B., and Guo, L. (2017). Knowledge graph embedding: A survey of approaches and applications. *IEEE Trans. Knowl. Data Eng.*, 29(12):2724–2743.

West, R., Gabrilovich, E., Murphy, K., Sun, S., Gupta, R., and Lin, D. (2014). Knowledge base completion via search-based question answering. In Chung, C., Broder, A. Z., Shim, K., and Suel, T., editors, *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014*, pages 515–526. ACM.

Wu, H., Fujiwara, T., Yamamoto, Y., Bolleman, J. T., and Yamaguchi, A. (2014). BioBenchmark Toyama 2012: An evaluation of the performance of triple stores on biological data. *J. Biomed. Semant.*, 5:32.

Xiong, B., Potyka, N., Tran, T., Nayyeri, M., and Staab, S. (2022). Box embeddings for the description logic EL++. *CoRR*, abs/2201.09919.

Yang, B., Yih, W., He, X., Gao, J., and Deng, L. (2015a). Embedding entities and relations for learning and inference in knowledge bases. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Yang, B., Yih, W., He, X., Gao, J., and Deng, L. (2015b). Embedding entities and relations for learning and inference in knowledge bases. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Zhang, S., Tay, Y., Yao, L., and Liu, Q. (2019). Quaternion knowledge graph embeddings. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 2731–2741.

Zhang, Y., Dai, H., Kozareva, Z., Smola, A. J., and Song, L. (2018). Variational reasoning for question answering with knowledge graph. In McIlraith, S. A. and Weinberger, K. Q., editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 6069–6076. AAAI Press.

Zhang, Z., Cai, J., Zhang, Y., and Wang, J. (2020). Learning hierarchy-aware knowledge graph embeddings for link prediction. 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 3065–3072. AAAI Press.

Zhang, Z., Wang, J., Chen, J., Ji, S., and Wu, F. (2021). Cone: Cone embeddings for multi-hop reasoning over knowledge graphs. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 19172–19183.

Zheng, W., Wang, W., Qian, F., Zhao, S., and Zhang, Y. (2022). Hyperbolic hierarchical knowledge graph embeddings for link prediction in low dimensions. *CoRR*, abs/2204.13704.

Zhu, Z., Zhang, Z., Xhonneux, L. A. C., and Tang, J. (2021). Neural bellman-ford networks: A general graph neural network framework for link prediction. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 29476–29490.