PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE
SCHOOL OF ENGINEERING

BUILDING A QUERY LANGUAGE FOR THE WEB OF DATA: EFFICIENCY IN AN OPEN WORLD

MARTÍN I. UGARTE C.

Thesis submitted to the Office of Graduate Studies
in partial fulfillment of the requirements for the degree of
Doctor in Engineering Sciences

Advisor:
MARCELO ARENAS S.

Santiago de Chile, September, 2015

© 2015, Martín Ugarte Caraball

PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE
SCHOOL OF ENGINEERING

BUILDING A QUERY LANGUAGE FOR THE WEB OF DATA: EFFICIENCY IN AN OPEN WORLD

MARTÍN I. UGARTE C.

Members of the Committee:

MARCELO ARENAS S.
CLAUDIO GUTIÉRREZ
JORGE PÉREZ R.
JORGE BAIER A.
ÓSCAR CORCHO G.
JORGE VÁSQUEZ

Thesis submitted to the Office of Graduate Studies
in partial fulfillment of the requirements for the degree of
Doctor in Engineering Sciences

Santiago de Chile, September, 2015

© 2015, MARTÍN UGARTE CARABALL

To those who stand against
intellectual property

ACKNOWLEDGEMENTS

I would like to thank, in no particular order:

Marcelo, for being a great advisor and an even better person. For taking the time to discuss every time I stopped by his office and for patiently listening to my (not always good) ideas. There is absolutely nothing bad I could say about my experience as his student, and I'm sincerely grateful for everything he has done for me.

Juan Reutter and Jorge Baier, for always taking the time to listen, think and provide the right advise. Was it about beer, logics or human beings.

Soledad, for her immense affection and invaluable administrative help. My experience as a graduate student was really pleasant because of her.

Luis Dissett, for showing me a discrete path in a world of continuum.

Felipe, Benjamín and Raimundo, the former brightest students in my class and today my friends, for believing in me and taking the time to develop ideas together.

Everyone at CSWR, the most cheerful research group I have ever come across.

Diego, Ignacio B., Aníbal and Ignacio R., for sharing with me their enlightening lives and thoughts. They are men of my own heart.

The great "boys and girls", for being the best (and most stupid) group of friends that I could have ever asked for.

Pedro, José, Jaime and their families, for always considering me as one of their own.

Alejandra, for sharing with me her love and joy during these years. For always having time for me and for her support in everything I do.

My parents, my good brother, my sister and my little sister, for their unconditional love and support. Without a doubt you are the best family. This work is for you.

My doctoral studies were partially funded by CONICYT Doctorate's scholarship CONICYT-PCHA - 21120368 and the Millennium Nucleus Center for Semantic Web Research under Grant NC120004.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE
ESCUELA DE INGENIERIA

CONSTRUYENDO UN LENGUAJE DE CONSULTAS PARA LA WEB DE DATOS: EFICIENCIA EN UN MUNDO ABIERTO

Tesis enviada a la Dirección de Postgrado en cumplimiento parcial de los requisitos para el grado de Doctor en Ciencias de la Ingeniería

MARTÍN IGNACIO UGARTE CARABALL

RESUMEN

Al consultar bases de datos incompletas es importante tener la posibilidad de extender los resultados cuando hay información adicional disponible. Esta funcionalidad ha sido ampliamente adoptada para consultar la Web Semántica, dado que la información en la Web es inherentemente incompleta. Desafortunadamente, la implementación de esta funcionalidad en SPARQL, el lenguaje recomendado por el World Wide Web Consotrium (W3C) para consultar datos en la Semantic Web, trae consigo algunos efectos negativos. Dos de los más importantes son un incremento en la complejidad de evaluar consultas, y un conflicto entre SPARQL y el supuesto de mundo abierto. Diferentes caminos han sido tomados para arreglar estos problemas, de los cuales el más adoptado ha sido restringir SPARQL a patrones *bien-diseñados*. Sin embargo, aun sigue abierto el problema de determinar si éste es el enfoque correcto en términos de complejidad de evaluación y poder expresivo.

El punto de partida de esta tesis el estudio de las propiedades fundamentales que debiese satisfacer un lenguaje de consultas para la Web Semántica, en particular considerando que la información es incompleta. Para esto, investigamos las técnicas que han sido desarrolladas en la lógica de primer orden para caracterizar sintácticamente propiedades semánticas. Luego presentamos un marco teórico que nos permite aplicar estas técnicas al caso de SPARQL, definiendo lenguajes que resultan naturales para capturar las propiedades deseadas. También estudiamos las propiedades computacionales de estos lenguajes para entender su aplicabilidad en implementaciones del mundo real.

Lo primero que hacemos es mostrar que los enfoques adoptados anteriormente no son suficientes para conciliar la semántica de SPARQL con el hecho de que la información en la Web es incompleta. Luego definimos un nuevo operador para obtener información opcional, el cual nace naturalmente de las técnicas que estudiamos en lógica de primer orden. Este operador nos permite definir fragmentos de SPARQL con buenas propiedades en términos de poder expresivo y complejidad de evaluación. Luego, nos enfocamos en consultas SPARQL de tipo CONSTRUCT, que son aquellas que generan como resultado el mismo tipo de estructuras que reciben como entrada (grafos RDF). Bajo este fragmento somos capaces de definir un lenguaje de consultas que es simple y a la vez captura las nociones semánticas de interés. Por último, mostramos que este lenguaje presenta, sorprendentemente, una menor complejidad de evaluación que los fragmentos que han sido presentados anteriormente.

Palabras Clave: SPARQL, Web Semántica, Información Incompleta, Lenguajes de Consulta, Consultas CONSTRUCT, Complejidad Computacional

Miembros de la Comisión de Tesis Doctoral

Marcelo Arenas
Claudio Gutiérrez
Jorge Pérez R.
Jorge Baier A.
Óscar Corcho G.
Jorge Vásquez

Santiago, Septiembre, 2015

PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE
ESCUELA DE INGENIERIA

BUILDING A QUERY LANGUAGE FOR THE WEB OF DATA: EFFICIENCY IN AN OPEN WORLD

Thesis submitted to the Office of Graduate Studies in partial fulfillment of the requirements for the Degree of Doctor in Engineering Sciences by

MARTÍN I. UGARTE

ABSTRACT

When querying incomplete databases, a distinctive feature is the possibility of optionally extending the results if additional data is available. This feature has been widely adopted for querying the Semantic Web, given the inherent incompleteness of Web data. Unfortunately, its implementation in SPARQL, the language recommended by the World Wide Web Consotrium (W3C) for querying Semantic Web data, brings some negative effects. Two of the most notable are an increase in the complexity of evaluating queries, and a conflict between SPARQL and the underlying open-world assumption of Web data. Many approaches for fixing these problems have been proposed, being the most widely adopted that of restricting SPARQL to *well-designed* graph patterns. Nevertheless, the question of whether this is the right approach in terms of expressive power and evaluation complexity, remains an open problem since its introduction.

The starting point of this dissertation is the study of the fundamental properties that a language should have for querying incomplete data in the Semantic Web. We take a close look at techniques developed in first-order logic for characterizing semantic properties. We present a framework for applying these techniques to SPARQL, and for defining natural languages capturing the desired semantic properties. Furthermore, we study the computational properties of such languages in order to understand their applicability for real-world implementations.

We begin this dissertation by showing that previous approaches fall short in concealing the semantics of SPARQL with the incompleteness of Web data. Then we define a new operator for obtaining optional information, which naturally originates from applying techniques studied in first-order logic. This operator allows us to define fragments of SPARQL with novel properties in terms of expressive power and complexity of evaluation. Then, we focus on SPARQL CONSTRUCT queries, the set of queries in SPARQL that takes the same structures as input and output. Under this class of queries, we are able to define a clean language that precisely captures the desired semantic notions, and, surprisingly, presents lower complexity of evaluation than fragments presented before.

Keywords: SPARQL, Semantic Web, Incomplete Information, Query Languages, CONSTRUCT queries, Computational Complexity

Members of the Doctoral Thesis Committee

Marcelo Arenas
Claudio Gutiérrez
Jorge Pérez R.
Jorge Baier A.
Óscar Corcho G.
Jorge Vásquez

Santiago, September, 2015

1. INTRODUCTION

The amount of information in the Web has grown considerably. Just by looking at the English Wikipedia we can find more than five million articles. The Wikileaks Foundation has released hundreds of thousands of documents in a single day. The amount of websites being served to the public has already surpassed one billion. The number of users on the Internet has recently reached three billions and, in average, each of them sends more than one email per day. And all this without considering advertising.

The information in the Web has grown without structure. There is plenty of information that, although available, demands immense human resources to be accessed. Tasks as simple as obtaining the list of Chilean researchers mentioned in Wikipedia are considerably hard. Our best chance is that someone else has already created that list, which does not mean that creating it did not take time. Moreover, the resources spent creating this list are specific to this particular case; they will be of no use even when creating the list of researchers in Wikipedia from a different country. With all the information available, the demand of human resources seems baffling, specially considering that the machines we carry in our pockets are capable of performing billions of operations and hundreds of thousands of memory accesses every second. But the fact that information cannot be easily accessed is not related to the lack of computational resources. Instead, the problem is that a good deal of data in the Web is unstructured, stored to be understood by humans and not by computers. If computers understand the information as we do, our everyday laptops would be capable of navigating and retrieving relevant information in a couple of seconds.

1.1. The Semantic Web

Tim Berners-Lee, one of the authors of the HTTP protocol and director of the World Wide Web Consortium (W3C), describes the lack of accessibility to data as *tremendously* *frustrating*. In 1998, there was already a discussion about a *Semantic WWW*, a World Wide Web with "machine-understandable information" (Berners-Lee, Hendler, Lassila, et al., 2001). The proposal was to build technology upon four basic principles, which gave birth to what is known today as Linked Open Data. The first and most important of these principles is to standardize the format for storing information across the Web. To this end, the W3C recommended the Resource Description Framework (RDF) (Manola & Miller, 2004), a data model designed to store Web resources. Under this framework, every resource (including concepts and relations) should have a unique identifier conforming to the HTTP protocol, in words of Berners-Lee, "one of those names starting with HTTP". These names would be related to each other by means of other names, because relationships should also have identifiers. Moreover, they should allow dereferencing, meaning that by only knowing the identifier of a resource one should be able to obtain all of its available relationships with other resources.

The Semantic Web did not take long to materialize. Individuals and institutions started publishing their data in RDF and adopting Linked Data technologies in the early 2000's. By 2013 more than four million Web domains were part of the Linked Open Data (Ramanathan, 2013). With the publication of information in the form of RDF came the problem of accessing all of this data. Several designs and proposals were presented for this purpose (see (Furche, Linse, Bry, Plexousakis, & Gottlob, 2006) for a survey). The one that received the most attention was SPARQL (Prud'hommeaux, Seaborne, et al., 2006), a SQL-flavoured query language for RDF data. SPARQL became a W3C standard in 2008 (Prud'hommeaux & Seaborne, 2008), and the current version (SPARQL 1.1) was issued in 2013 (W3C SPARQL Working Group, 2013). SPARQL is one of the key elements of the Semantic Web initiative, and it is actually recognized as part of the Semantic Web Stack (Figure 1.1). There are several SPARQL engines implemented available to industry and public use, e.g. (Seaborne, 2010; Erling & Mikhailov, 2009; Harris, Lamb, & Shadbolt, 2009).

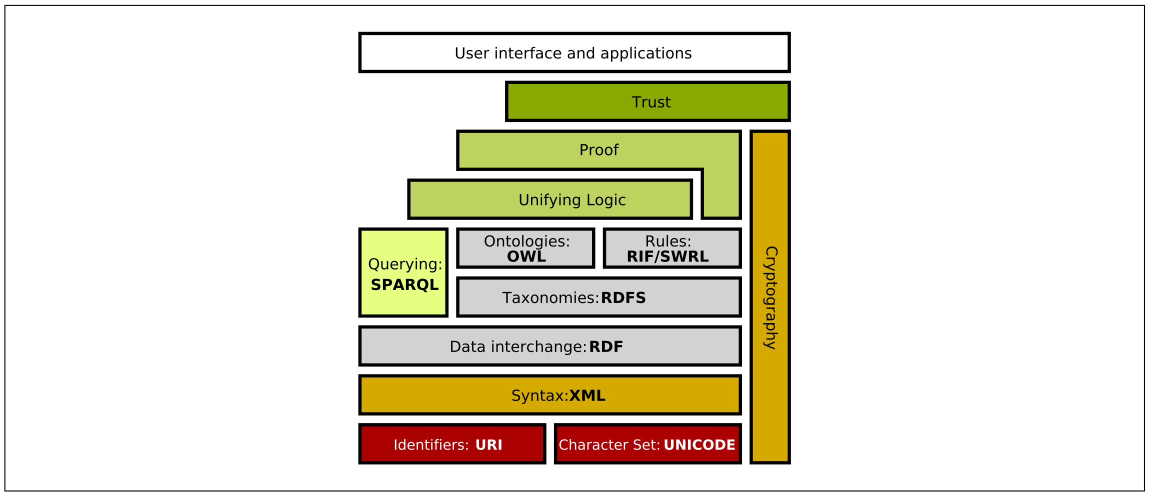

FIGURE 1.1. The Semantic Web Stack. SPARQL is recognized as a key element for accessing data stored as RDF.

1.2. Accessibility and performance

SPARQL was originally designed by looking at each desired feature in isolation, but it turned out to be a rather complicated language when all of these features were put together, presenting structural and performance problems. Aware of this situation, in (Pérez, Arenas, & Gutierrez, 2006a) the authors presented a formalization of the syntax and semantics of SPARQL. This work was seminal for the mathematical study of this language, and a good deal of research has been built upon this formalization. For example, studies about complexity of query evaluation (Schmidt, Meier, & Lausen, 2010; Losemann & Martens, 2012; Arenas, Conca, & Pérez, 2012; Picalausa & Vansummeren, 2011), query optimisation (Letelier, Pérez, Pichler, & Skritek, 2013; Pichler & Skritek, 2014; Chekol, Euzenat, Genevès, & Layaïda, 2012a, 2012b), federation (Buil-Aranda, Arenas, & Corcho, 2011), expressive power (Angles & Gutierrez, 2008a; Polleres & Wallner, 2013), and provenance tracking (Geerts, Karvounarakis, Christophides, & Fundulaki, 2013; Halpin & Cheney, 2014).

The formal study of SPARQL has impacted the Semantic Web community in several ways, influencing the form in which users query datasets and even the definition of the

SPARQL W3C standard. But despite the key importance of this language, we are still left with fundamental open problems that prevent the further adoption of the Semantic Web. In fact, the growth rate of the Linked Open Data has seen a drastic deceleration in past few years (Hogan & Gutierrez, 2014). This deceleration is attributed mainly to performance and availability issues.

Giving every user on the Web the power to query your infrastructure with a full-featured query language is a huge step towards data sharing, but has obvious performance consequences. The time needed to evaluate each query is of fundamental importance, as one small inefficiency could compromise a whole infrastructure when multiplied by a large number of users. In fact, this has been proved to be one of the key problems in the Semantic Web (Aranda, Hogan, Umbrich, & Vandenbussche, 2013). Therefore, it is natural to question whether SPARQL is the correct way of accessing information in the Semantic Web.

1.3. Incomplete information

Another important issue in the Semantic Web is that information in the Web is inherently incomplete. This is not the case, for example, in relational databases, where the information is considered complete and, therefore, unavailable information is assumed to be false. On the contrary, RDF was designed as a framework in which nothing is assumed about non-present data. This is known as the *open-world assumption* (OWA). When working over complete data, one knows exactly what are the properties of the entities, and therefore queries aim to retrieve specific properties. But this is not the case over incomplete data. For example, a user may want to retrieve a certain list of people, together with their nationalities. However, since data is incomplete, one could expect that not all nationalities are available. This does not prevent the user from wanting to obtain a complete list, including those people for whom nationalities are not known. Hence, the query for retrieving this information would be translated into natural language as "give me the list of people and, optionally, include the nationalities if they are available". This illustrates what is known as querying for *optional information*.

Unfortunately, there are SPARQL queries that behave contrary to the OWA, and hence contrary to the design of RDF. This again rises the question of whether SPARQL is the correct language for querying the Semantic Web, and offers a negative answer. This problem has been previously studied from a general perspective. In (Pérez, Arenas, & Gutierrez, 2009), the authors identify a high-level condition that is satisfied precisely by queries that conform to the OWA. This is the notion of *weak-monotonicity*. Essentially, a query is weakly-monotone if whenever new data is added to a dataset, the evaluation of such query returns at least as much information as it did before. For example, the query presented above that asks for a list of people and their nationalities is weakly-monotone: Assume that we evaluate this query over a dataset, and obtain a certain list of people. Some of the answers contain just a person, and some contain a person and a nationality. Suppose now that new information is added to the dataset, and the query is evaluated again. Maybe new answers will appear on the list, and maybe nationalities will be added to answers that only contained a person, but no information will be lost. This implies that the query is weakly-monotone, which intuitively corresponds to the idea of preserving the information under data extensions.

Weak-monotonicity is also important as it prevents users from querying for *negative* information. For example, consider a query that asks for "the list of people that are not Chilean". To answer this query, it is necessary to make an assumption over unavailable data, as the nationality of some people is not known. We conclude that this query cannot be answered properly over the OWA. Negative queries are not weakly-monotone, showing that weak-monotonicity is a fundamental concept for querying RDF data. This is reinforced by the fact that weakly-monotone queries preserve information under data extensions, property that has proven to be fruitful in several areas of databases. For example, in the context of data exchange, queries preserving information under data extensions allow for a clean definition of the semantics (Libkin, 2006; Hernich, Libkin, & Schweikardt, 2011; Libkin & Sirangelo, 2011), and the same occurs in the context of data integration (Fagin, 1996; Lenzerini, 2002; Abiteboul & Duschka, 1998).

The notion of weak-monotonicity does not provide any insight about how to effectively write weakly-monotone queries. Moreover, one can show that there is no computer program that can decide, given a SPARQL query, whether or not such query is weakly-monotone. Hence, the problem of finding a practical fragment of queries that satisfies this condition is instrumental in the search for a correct way of accessing Semantic Web data. This problem has been addressed before, mainly by imposing certain restrictions on how to write SPARQL queries, but none of these restrictions have been proven to characterize the concept of weak-monotonicity. The most popular and adopted restrictions has been allowing only for well-designed queries (Pérez et al., 2009). This condition aims to prevent users from writing queries containing the aforementioned *inconsistencies* generated by the OPT operator.

Whether the fragment of SPARQL well-designed queries characterizes the set of weakly-monotone queries is still an open problem. However, well-designed queries are considered to be a good fragment in practice because they also present good properties in terms of evaluation efficiency. In terms of combined complexity (Vardi, 1982), evaluating a well-designed queries (CO-NP-complete when disallowing disjunction and projection, and Σ_2^p-coomplete in general (Letelier et al., 2013)) is easier than evaluating SPARQL queries in general (PSPACE-complete).

1.4. Learning from relational databases

The relational model has dominated the area of databases for more than twenty years (Abiteboul, Hull, & Vianu, 1995). The study of relational databases and its most popular query language, SQL, is an area of Computer Science on its own. Our understanding of the relational model relies on more than sixty years of research in mathematical logic, and is far more developed than our knowledge about the elements of the Semantic Web.

Therefore, it is natural to look at the techniques and design principles used in the relational model, and see whether they can be applied to the Semantic Web.

In SQL one answer to a query cannot contain more information than another answer to the same query. Thus, as opposed to SPARQL, preserving information over data extensions in the relational model is the same as preserving answer. The idea of preserving answers is known as monotonicity, and has played an important role in the development of areas like consistent query answering (Bertossi, 2006) and data exchange (Libkin, 2006). Just like in the case of weak-monotonicity, the problem of knowing if a query is monotone is undecidable; meaning that there is no computer program that receives a query as input, and can tell whether or not that query is monotone. Researchers have looked for reasonable restrictions that enforce queries to satisfy monotonicity. It turns out that theorems developed in the context of first-order logic in the late 50's can be applied to find such restrictions (Feferman, 2008). Moreover, these restrictions are easily verifiable, and are precise characterizations in the sense that a query satisfies the restriction if and only if it satisfies the semantic notion. For example, it is a well-known result that a first-order formula is monotone if and only if it is positive (Lyndon, 1959). Having this information, it is natural to study the fragments of SPARQL in which one query can not contain more information than another answer. Moreover, we could try to apply to SPARQL the same techniques used to find characterizations in first-order logic.

Another difference between SPARQL and SQL is the structural discrepancy between the input and the output of SPARQL queries. Evaluating an SQL query over a relational database results in another relational database, but evaluating a SPARQL query over an RDF dataset, in general, does not result in an RDF dataset. However, there is a fragment of SPARQL that generates RDF graphs as output, the fragment of CONSTRUCT queries. Given the structural similarity between CONSTRUCT queries and SQL it is conceivable that much more insight can be obtained by applying techniques from the relational model to this fragment. But rather surprisingly, and despite being an important part of the SPARQL standard, these queries have received almost no theoretical attention. Therefore we aim to study over CONSTRUCT queries the same problems about performance, accessibility and compliance to the OWA that were presented in the previous sections. It is also interesting to mention that, in the context of CONSTRUCT queries, the correct notion for capturing the queries conforming to the OWA is that of monotonicity. This presents another similarity with SQL, and therefore we can expect that results over the relational model apply more directly to the fragment of CONSTRUCT queries.

1.5. Hypothesis and goals

The main hypothesis of this dissertation is that the techniques developed in first-order logic over the past fifty years can be of use to find an appropriate language for querying the Semantic Web. More precisely, we hypothesize that by establishing a connection between first-order logic and SPARQL, we could use a methodology similar to that of proving preservation theorems by means of interpolation theorems to characterize the set of SPARQL queries conforming to the open-world semantics of RDF.

Our general goal is to find a proper query language for the Semantic Web, that addresses the aforementioned problems of high complexity of evaluation and compliance to the Open World Assumption. This derives on the following specific goals:

- To understand the techniques that have been developed in first-order logic for characterizing semantic properties similar to weak-monotonicity. In particular, to understand how are interpolation results used to prove Lyndon's positivity theorem (Lyndon, 1959) and the Łoś-Tarski theorem (Hodges, 1997).
- To build a framework that establishes a connection between first-order logic and SPARQL. This framework must allow us to apply the interpolation techniques used in first-order logic to the case of SPARQL.
- To study what are the implications of applying interpolation to SPARQL. In particular, to characterize languages satisfying the semantic properties required by the open-world assumption.

- To understand if the defined fragments are suitable for real-world applications, by studying the computational complexity of evaluating queries in those fragments.

1.6. Summary of contributions

1.6.1. Well-designed graph patterns

We start by studying the fragment of SPARQL well-designed graph patterns including disjunction at the top-most level, and whether this fragment is capable of capturing every query conforming to the OWA and to the principles of RDF. We show that this is not the case by providing a simple query that is weakly-monotone but that can not be equivalent to any well-designed graph pattern. This presents a big advance in our understanding of SPARQL and motivates the search for new ways of querying RDF data for optional information.

1.6.2. A framework for the study of weak-monotonicity

We present a thorough and comprehensible introduction to Lyndon's interpolation theorem (Lyndon, 1959) and Otto's interpolation theorem (Otto, 2000). We show in detail how these results are applied to prove preservation theorems. By showing a correspondence between SPARQL and first-order logic, we provide a framework for applying interpolation theorems to prove characterizations of weakly-monotone fragments of SPARQL.

It is important to mention that interpolation theorems are known to be valid under *arbitrary* (finite and infinite) models. This implies that the characterization results derived from interpolation only apply to queries that satisfy a semantic notion when considering arbitrary models. To illustrate this difference, consider the characterization of monotonicity in first-order logic. There are formulas that are monotone when restricting the structures to finite models, but that are not monotone in general. These formulas are not characterized by this theorem. This has to be considered in the case of the Semantic Web as the definition of RDF applies to real-world datasets, which are finite collections. Hence, if we obtain a characterization of weak-monotonicity using the presented framework, it would leave out those weakly-monotone queries that are not weakly-monotone when extending to arbitrary databases. Nevertheless, it is well-known that finding queries that satisfy a condition over finite datasets but not under arbitrary datasets are not expected to derive from real-world applications (Ajtai & Gurevich, 1987). Therefore, the framework we develop is a valid tool for obtaining practical applications for RDF and SPARQL.

1.6.3. Maximal answers in SPARQL

We prove that given a weakly-monotone graph pattern P, there is a graph pattern Q that does not mention the operator for obtaining optional information (OPT), and that is equivalent to P in the set of *maximal* answers. Maximal answers are answers that contain information that is not present in any other answer. We define a new operator called Not-Subsumed used to obtain the maximal answers from a graph pattern. We prove that the fragment of weakly-monotone graph patterns that only output maximal answers is characterized by the set of patterns in which OPT is not mentioned, and Not-Subsumed is only allowed at the top-most level. Graph patterns in this fragment are denoted simple patterns. We also study the fragment of disjunctions of weakly-monotone graph patterns that only output maximal answers. We prove that this fragment is, as expected, captured by the fragment of disjunctions of simple patterns. Graph patterns in this fragment are denoted ns-patterns.

Unfortunately, we are not able to deduce that ns-patterns cover the set of weakly-monotone graph patterns. This derives from the application of interpolation techniques, which is restrictive in preserving the answers that are not maximal. The problem of knowing if the set of ns-patterns captures the fragment of all weakly-monotone graph patterns remains open.

1.6.4. Monotonicity and CONSTRUCT queries

The need for maximal answers derives directly from the application of interpolation, but rely significantly on the fact that the evaluation of a SPARQL graph pattern over a dataset does not produce an RDF dataset as output. Therefore we study a different fragment of queries for RDF, namely CONSTRUCT queries. Although these queries have been part of the official standard of SPARQL since its original definition, they have received very little theoretical attention. We formalize this fragment and show that it has practical advantages over other fragments of SPARQL. One of these advantages is that monotonicity is the correct semantic notion for capturing CONSTRUCT queries that conform to the OWA. We show that, surprisingly, the fragment of monotone CONSTRUCT queries is characterized by the set of CONSTRUCT queries in which only the operators CONSTRUCT, AND, UNION and FILTER are allowed. Queries in this fragment are denoted AUF CONSTRUCT queries.

We define the fragment of well-designed CONSTRUCT queries, and we prove that it is also equivalent to the fragment of AUF CONSTRUCT queries. We show an effective transformation from well-designed CONSTRUCT queries to AUF CONSTRUCT queries. As this equivalence obviously holds when restricting to finite RDF graphs, we conclude that well-designed CONSTRUCT queries precisely captures the set of CONSTRUCT queries that are monotone under arbitrary RDF datasets.

1.6.5. Computational complexity

We analyze how suitable are the found fragments for real-world applications by studying their evaluation combined complexity (Vardi, 1982). We show that the evaluation problem for simple patterns is DP-complete. Then, we show that if we allow for a fixed number of disjuncts in ns-patterns, the evaluation problem falls in different levels of the boolean hierarchy (Wechsung, 1985). Specifically, if the amount of simple patterns in an ns-pattern is restricted to a natural number k, then the evaluation problem turns out to be complete for the class BH_{2k}, the $2k^{\text{th}}$ level of the boolean hierarchy. Then, we study the evaluation complexity of ns-patterns with an arbitrary amount of disjuncts. We prove that this problem is complete for the complexity class $\mathrm{P}_{\|}^{\mathrm{NP}}$. This class contains all languages that can be solved in polynomial time by a Turing machine that can issue a polynomial amount of queries in parallel to an NP oracle. Finally, we show that the complexity of evaluating AUF CONSTRUCT queries is NP-complete. From this result, and from the fact that AUF CONSTRUCT queries characterize monotonicity, we conclude that AUF CONSTRUCT queries are a proper language for querying RDF data.

2. PRELIMINARIES

We start by presenting the mathematical foundations of the Semantic Web. We follow the algebraic formalization presented in (Pérez, Arenas, & Gutierrez, 2006b) extended with projection. To refer to different fragments of SPARQL we borrow notation from (Schmidt et al., 2010).

2.1. RDF

The Resource Description Framework is based upon *identifiers* representing Web resources. To allow for constant values (like numbers or strings) and existential values (resources with unknown identifiers), literal values and blank nodes, respectively, are included in the framework. Assume there are infinite pairwise disjoint sets $\mathbf{I}$, $\mathbf{B}$ and $\mathbf{L}$. $\mathbf{I}$ is a set of International Resource Identifiers (IRIs) (Durst & Suignard, 2005), $\mathbf{B}$ is a set of Blank Nodes (Mallea, Arenas, Hogan, & Polleres, 2011), and $\mathbf{L}$ is a set of Literal Values. A triple $(s, p, o) \in (\mathbf{I} \cup \mathbf{B}) \times \mathbf{I} \times (\mathbf{I} \cup \mathbf{B} \cup \mathbf{L})$ is called an RDF triple, and represents that the subject s is related to the object o by the predicate p. An RDF dataset is simply a finite collection of RDF triples. Throughout this document we disallow blank nodes and literal values, considering only the set $\mathbf{I}$ and RDF triples in $\mathbf{I}\times\mathbf{I}\times\mathbf{I}$. Most of the results presented in this dissertation are not affected by this assumption. Whenever this is not the case it will be explicitly stated. We also assume, for the sake of simplicity, that every string can be used an identifier.

Example 1. *Assume that we want to store information about the founders and supporters of an organization. We need to state every relationship as an RDF triple (e.g., if a person s founded an organization o, we store the triple* $(s, founder, o)$*). Table 2.1 displays such an RDF dataset. Notice that a resource mentioned in one triple as a property can also be mentioned in another triple as subject or predicate, e.g.,* founder *and* supporter.

TABLE 2.1. Tabular representation of an RDF dataset. All resources can be mentioned as subjects, predicates or objects.

Subject	**Predicate**	**Object**
Gottfrid Svartholm	founder	The Pirate Bay
Fredrik Neij	founder	The Pirate Bay
Peter Sunde	founder	The Pirate Bay
founder	sub_property	supporter
The Pirate Bay	stands_for	sharing rights
Carl Lundström	supporter	The Pirate Bay

The previous example illustrates how to use RDF triples to store data. However, as RDF triples are relations between entities, it is natural to represent RDF datasets as graphs. Figure 2.1 represents the same dataset as Table 2.1. We indistinctly refer to RDF datasets by RDF *graphs*.

2.2. SPARQL syntax

Queries in the core fragment of SPARQL are called graph patterns, given SPARQL is essentially a pattern-matching query language for RDF graphs. To define the set of

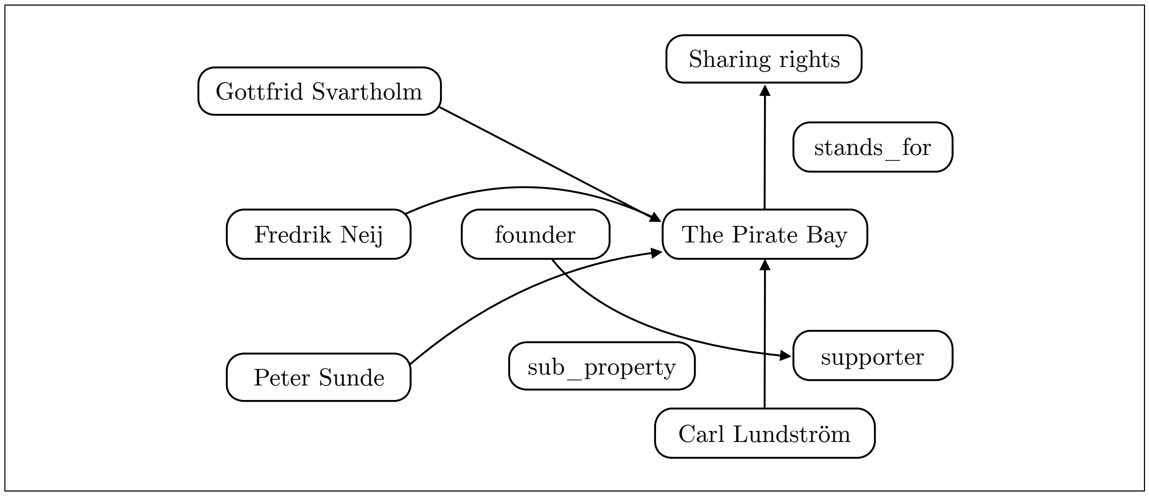

FIGURE 2.1. Graph representation of an RDF dataset. Each edge starts on a node s, passes close to a node p, and points to a node o, representing the triple (s, p, o).

SPARQL graph patterns, assume there is an infinite set $\mathbf{V}$ of variables, disjoint from $\mathbf{I}$. Elements of $\mathbf{V}$ are distinguished by using ? as a prefix (e.g. $?X$ is a variable in $\mathbf{V}$).

DEFINITION 2.2.1. *The set of SPARQL graph patterns is recursively defined as follows:*

- *A triple in* $(\mathbf{I} \cup \mathbf{V}) \times (\mathbf{I} \cup \mathbf{V}) \times (\mathbf{I} \cup \mathbf{V})$ *is a graph pattern (called a triple pattern).*
- *If* P_1*,* P_2 *are graph patterns, then* $(P_1 \text{ UNION } P_2)$*,* $(P_1 \text{ AND } P_2)$*,* $(P_1 \text{ OPT } P_2)$ *are graph patterns.*
- *If P is a graph pattern and V is a finite subset of* $\mathbf{V}$*, then* $(\text{SELECT } V \text{ WHERE } P)$ *is a graph pattern.*
- *If P is a graph pattern and R is a SPARQL built-in condition, then* $(P \text{ FILTER } R)$ *is a graph pattern.*

A SPARQL built in condition is a propositional formula where atoms are equalities or inequalities over the set $\mathbf{I} \cup \mathbf{V}$ plus other features (Prud'hommeaux & Seaborne, 2008). We restrict to the fragment of built-in conditions presented in (Pérez et al., 2006a), which is formally defined as follows:

- If $?X, ?Y \in \mathbf{V}$ and $c \in \mathbf{I}$, then bound$(?X)$, $?X = c$, $?X = ?Y$ are built in-conditions.
- If R_1 and R_2 are built-in conditions, then $(\neg R_1)$, $(R_1 \vee R_2)$ and $(R_1 \wedge R_2)$ are built-in conditions.

If O is an operator in {UNION, AND, OPT, FILTER, SELECT}, we say that a graph pattern P is O-free if O does not occur in P. To refer to a fragment of SPARQL in which only some operators are allowed, we use the first letter of AND, UNION, OPT and FILTER. For the SELECT operator we use a superscript π. For example AUOF represents the fragment of SELECT-free graph patterns, while AF^{π} is the fragment of graph patterns constructed by using AND, FILTER and SELECT. Having defined the set of SPARQL graph patterns, we proceed to define their evaluation over RDF graphs.

2.3. Semantics of SPARQL

To define the semantics of SPARQL we need to recall some further notation. If P is a graph pattern, var(P) denotes the set of all variables mentioned in P, and $\mathbf{I}(P)$ is the set of all IRIs mentioned in P. A mapping μ is a partial function $\mu : \mathbf{V} \to \mathbf{I}$. The domain of μ is the subset of $\mathbf{V}$ where μ is defined, and is denoted by dom(μ). The image of a mapping μ is denoted by range(μ). Given a mapping μ and a triple pattern t such that $\text{var}(t) \subseteq \text{dom}(\mu)$, $\mu(t)$ is the result of replacing, for every $?X \in var(t)$, every occurrence of $?X$ by $\mu(?X)$. A mapping μ_1 is said to be compatible with a mapping μ_2, denoted by $\mu_1 \sim \mu_2$, if for every $?X \in \text{dom}(\mu_1) \cap \text{dom}(\mu_2)$ it is the case that $\mu_1(?X) = \mu_2(?X)$. In this case, $\mu_1 \cup \mu_2$ denotes the extension of μ_1 to the variables in $\text{dom}(\mu_2) \setminus \text{dom}(\mu_1)$ according to μ_2. If two mappings μ_1 and μ_2 are not compatible we write $\mu_1 \nsim \mu_2$.

Let Ω_1 and Ω_2 be two sets of mappings. Then the join of, union of, difference between, and left-outer join of Ω_1 and Ω_2 are defined, respectively, as follows:

$$\begin{aligned}
\Omega_1 \bowtie \Omega_2 &= \{\mu_1 \cup \mu_2 \mid \mu_1 \in \Omega_1, \mu_2 \in \Omega_2 \text{ and } \mu_1 \sim \mu_2\} \\
\Omega_1 \cup \Omega_2 &= \{\mu \mid \mu \in \Omega_1 \text{ or } \mu \in \Omega_2\} \\
\Omega_1 \setminus \Omega_2 &= \{\mu \in \Omega_1 \mid \text{ for all } \mu' \in \Omega_2 \text{ it is the case that } \mu \nsim \mu'\} \\
\Omega_1 ⟕ \Omega_2 &= (\Omega_1 \bowtie \Omega_2) \cup (\Omega_1 \setminus \Omega_2).
\end{aligned}$$

Finally, given a mapping μ and a set $V \subseteq \mathbf{V}$, the expression $\mu_{|V}$ represents the mapping that results from restricting μ to $\text{dom}(\mu) \cap V$.

We have now the necessary notions for defining the answer to a SPARQL graph pattern over an RDF graph G. The formal semantics of SPARQL is defined as follows:

DEFINITION 2.3.1. *Let G be an RDF graph and let P be a SPARQL graph pattern. The evaluation of P over G is denoted by $[\![P]\!]_G$, and is a set of mappings recursively defined as follows:*

- *If P is a triple pattern then $[\![P]\!]_G = \{\mu \mid dom(\mu) = var(t)$ and $\mu(t) \in G\}$*
- *If P is $(P_1 \text{ AND } P_2)$ then $[\![P]\!]_G = [\![P_1]\!]_G \bowtie [\![P_2]\!]_G$*
- *If P is $(P_1 \text{ OPT } P_2)$ then $[\![P]\!]_G = [\![P_1]\!]_G \rtimes\!\!\!\bowtie [\![P_2]\!]_G$*
- *If P is $(P_1 \text{ UNION } P_2)$ then $[\![P]\!]_G = [\![P_1]\!]_G \cup [\![P_2]\!]_G$*
- *if P is* SELECT V WHERE P' *then* $[\![P]\!]_G = \{\mu_{|V} \mid \mu \in [\![P']\!]_G\}$
- *if P is P'* FILTER R *then* $[\![P]\!]_G = \{\mu \mid \mu \in [\![P']\!]_G$ *and* $\mu \models R\}$

Given a mapping μ and a built-in condition R, we say that μ satisfies R, denoted by $\mu \models R$, if one of the next conditions hold:

- R is bound$(?X)$ and $?X \in \text{dom}(\mu)$
- R is $?X = c$, $?X \in \text{dom}(\mu)$ and $\mu(?X) = c$
- R is $?X = ?Y$, $?X, ?Y \in \text{dom}(\mu)$ and $\mu(?X) = \mu(?Y)$
- R is $(R_1 \vee R_2)$ and $\mu \models R_1$ or $\mu \models R_2$
- R is $(R_1 \wedge R_2)$ and $\mu \models R_1$ and $\mu \models R_2$
- R is $(\neg R')$ and $\mu \not\models R'$.

Two graph patterns P_1 and P_2 are said to be equivalent, denoted by $P_1 \equiv P_2$, if for every RDF graph G it is the case that $[\![P_1]\!]_G = [\![P_2]\!]_G$. Given a mapping μ with $\text{dom}(\mu) = \{?X_1, \ldots, ?X_n\}$, we sometimes refer to μ by $[?X_1 \to \mu(?X_1), \ldots, ?X_n \to \mu(?X_n)]$.

The following example illustrates the syntax and semantics of SPARQL.

Example 2. *Let G be the RDF graph illustrated in Figure 2.1. Assume we want to query for all the founders and supporters of organizations that stand for sharing rights. This would be achieved by the following graph pattern:*

$$P = \text{SELECT } \{?p\} \text{ WHERE } [(?o, \text{stands\_for}, \text{sharing rights}) \text{ AND } ((?p, \text{founder}, ?o) \text{ UNION } (?p, \text{supporter}, ?\text{o}))]$$

The evaluation of P over G is performed in a bottom-up fashion. We first evaluate the triple patterns, obtaining the sets of mappings shown in Figure 2.2. Then, from the UNION

$[\![(?o, \mathit{stands\_for}, \mathit{sharing\ rights})]\!]_G =$ (a)

?o
The Pirate Bay

$[\![(?p, \mathit{founder}, ?o)]\!]_G =$ (b)

?p	?o
Gottfrid Svartholm	The Pirate Bay
Fredrik Neij	The Pirate Bay
Peter Sunde	The Pirate Bay

$[\![(?p, \mathit{supporter}, ?o)]\!]_G =$ (c)

?p	?o
Carl Lundström	The Pirate Bay

FIGURE 2.2. The evaluation of a SPARQL graph patterns starts with the evaluation of triple patterns.

pattern we obtain the mappings in both (b) and (c). These mappings are matched against the only mapping in (a), generating the result of the AND pattern. In this particular case, the evaluation of the AND pattern also contains the mappings in (b) and (c). Then, the SELECT *operator projects the mappings to* $?p$*, generating the desired list of people.*

Although each of the defined operators might seem simple when looked separately, SPARQL turns to be a very powerful and complex query language. Moreover, the OPT operator can introduce some problems when combined in certain ways with other operators. In the next chapter we formally introduce these problems, and show how previous attempts to solve them fail in some particular cases.

3. INCOMPLETE INFORMATION IN THE SEMANTIC WEB

In this chapter we discuss the main issue of this dissertation, namely the conflict between SPARQL and the open-world semantics of RDF. To this end, we first recall the definition of monotonicity and weak-monotonicity. Then, we present the definition of well-designed patterns, which is one of the most adopted fragments for querying RDF graphs. We analyze whether this approach is the correct query language for SPARQL, and prove that the fragment of well-designed graph patterns is not expressive enough to capture weak-monotonicity.

3.1. The Open-World Assumption in SPARQL

Intuitively, a query language is said to conform to the Open-World Assumption if its semantics is defined in a way such that, when evaluating queries, nothing is assumed about non-present information. This is particularly important when considering information over the Web, where it is common to find incomplete data. The formal notion that captures this intuition for relational databases, First Order Logic, Datalog, and several other database formalisms, is that of monotonicity. Essentially, a query is said to be monotone if, whenever it outputs an answer over a dataset, it outputs that same answer over all extensions of that dataset. The formal definition of this concept over SPARQL is as follows:

DEFINITION 3.1.1. *Let P be a SPARQL graph pattern. P is said to be monotone if for every two RDF graphs G_1 and G_2 such that $G_1 \subseteq G_2$, it is the case that $[\![P]\!]_{G_1} \subseteq [\![P]\!]_{G_2}$.*

However, and contrary to the previously mentioned formalisms, monotonicity over SPARQL does not capture the desired property of concealing the semantics with the Open-World Assumption. This occurs precisely because SPARQL graph patterns allow for optional information, and hence one answer (mapping) can contain *more information* than another answer. This does not occur, for example, in relational databases, where every

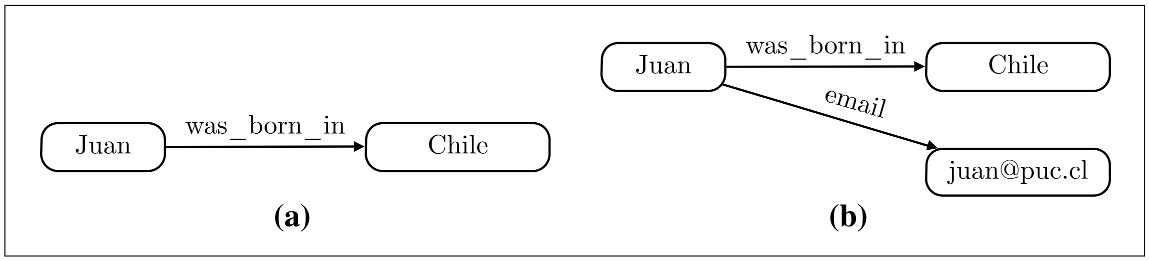

FIGURE 3.1. Graph (b) matches optional information that is not matched by graph (a) when evaluating $(?X, was\_born\_in, Chile)$ OPT $(?X, email, ?Y)$.

answer is an atomic piece of information. The next example illustrates that this is not the case for SPARQL.

Example 3. *Consider the graph pattern*

$$P = (?X, was\_born\_in, Chile) \text{ OPT } (?X, email, ?Y).$$

Let G_1 and G_2 be the RDF graphs (a) and (b) illustrated in Figure 3.1, respectively. Notice that G_2 is an extension of G_1 by adding the triple $(juan, email, juan@puc.cl)$. On one hand, the answer to P over G_1 contains only the mapping $\mu_1 = [?X \to juan]$, as $(?X, email, ?Y)$ does not match any triple in G_1. On the other, the evaluation of P over G_2 also contains one mapping: $\mu_2 = [?X \to juan, ?Y \to juan@puc.cl]$. We can conclude that the pattern P is not monotone, as the mapping μ_1 is not present in the answer to P over G_2. However, all the information in μ_1 can be retrieved from μ_2, and hence we can safely say that no information was lost when evaluating P over the extension of the original dataset.

In the previous example, it is clear that no assumption was made about non-present information when evaluating the graph pattern over the first dataset: the mapping $[?X \to juan]$ is part of the answer because the pattern $(?X, was\_born\_in, Chile)$ matches the triple $(juan, was\_born\_in, Chile)$, and does not rely on the occurrence of other triples in the dataset. Intuitively, this implies that P does conform to the incompleteness of Web data.

We conclude that monotonicity is not the correct notion capturing the desired set of graph patterns.

In (Pérez et al., 2009), the authors addressed this issue and introduced the notion of *weak-monotonicity*, a semantic condition intended to capture the graph patterns conforming to the open-world semantics of RDF. We proceed to recall the formal definition of this notion. Weak-monotonicity is defined in terms of *subsumption* of mappings. Given two mappings μ and μ', the mapping μ is said to be subsumed by μ', denoted by $\mu \preceq \mu'$, if it is the case that $\text{dom}(\mu) \subseteq \text{dom}(\mu')$ and $\mu(?X) = \mu'(?X)$ for every $?X \in \text{dom}(\mu)$. We say that μ is properly subsumed by μ' (denoted by $\mu \prec \mu'$) if $\mu \preceq \mu'$ and $\mu' \neq \mu$. To deal with answers to SPARQL graph patterns, these definitions are extended to sets of mappings. Given two sets of mappings Ω_1 and Ω_2, Ω_1 is said to be subsumed by Ω_2, denoted by $\Omega_1 \sqsubseteq \Omega_2$, if for every mapping $\mu_1 \in \Omega_1$, there is a mapping $\mu_2 \in \Omega_2$ such that $\mu_1 \preceq \mu_2$. Now we recall the definition of weak-monotonicity, which will play a key role throughout the next chapters of this dissertation.

DEFINITION 3.1.2. *Let P be a SPARQL graph pattern. P is said to be weakly-monotone if for every two RDF graphs G_1 and G_2 such that $G_1 \subseteq G_2$, it is the case that $[\![P]\!]_{G_1} \sqsubseteq [\![P]\!]_{G_2}$.*

As discussed earlier, in the answer to a graph pattern the atomic piece of information is not a mapping, but an assignation of a variables to an IRI. Hence, it is not hard to see that the definition above correctly captures the property of not losing information when new data is added to a dataset. In particular, notice that the non-monotone graph pattern of Example 3 is indeed weakly-monotone.

3.2. SPARQL and incomplete information

In the previous section we discussed the open-world semantics over RDF, and how the Semantic Web takes into consideration that data in the Web is incomplete. We also recalled the definition of weak-monotonicity and explained why it correctly captures the desired property of making no assumptions about unknown information. Unfortunately (and somewhat surprisingly), there are SPARQL graph patterns are not weakly-monotone. Consider the following example:

Example 4. *Let P be the graph pattern defined as*

$$P = (?X, awas\_born\_in, Chile) \text{ AND } ((?Y, was\_born\_in, Chile) \text{ OPT } (?Y, email, ?X))$$

Let G_1 and G_2 be the RDF graphs (a) and (b) of Figure 3.1, respectively. When evaluating P over G_1, both triple patterns $(?X, awas\_born\_in, Chile)$ and $(?Y, awas\_born\_in, Chile)$ matches the triple $(Juan, was\_born\_in, Chile)$. As the pattern $(?Y, email, ?X)$ does not match any triple in G_1, we obtain $[\![P]\!]_{G_1} = \{[?X \to Juan], [?Y \to Juan]\}$. If we evaluate P over G_2, we obtain the same results from the triples mentioning the IRI was_born_in. But now the triple $(?Y, email, ?X)$ matches $(Juan, email, juan@puc.cl)$, and hence we obtain

$$[\![(?Y, was\_born\_in, Chile) \text{ OPT } (?Y, email, ?X)]\!]_{G_2} = \{[?Y \to Juan, ?X \to juan@puc.cl]\}.$$

As $[\![(?X, was\_born\_in, Chile)]\!]_{G_2} = [?X \to Juan]$, the mappings coming from the two sides of the AND are not compatible. We conclude that $[\![P]\!]_{G_2} = \emptyset$. This implies that P is not weakly-monotone: $G_1 \subseteq G_2$ but $[\![P]\!]_{G_1} \nsubseteq [\![P]\!]_{G_2}$.

In the example above, when evaluating P over G_1 the triple $(2, c, d)$ was assumed to be a false piece of information. As shown in the evaluation over G_2, if this triple represented a true relation then the mapping $[?X \to 1, ?Y \to 1]$ would have not been in $[\![P]\!]_{G_1}$. This example shows that there are SPARQL graph patterns that are not weakly-monotone, which intuitively conform neither to the open-world semantics of RDF nor to the incompleteness of Web data.

The graph pattern P of Example 4 is somewhat unnatural. The triple $(?Y, email, ?X)$ offers optional information to $(?Y, was\_born\_in, Chile)$, but at the same time is intended to match the results of $(?X, awas\_born\_in, Chile)$. Therefore, whenever $(?Y, email, ?X)$ does not match any triple in the graph, every mapping in the evaluation of the OPT pattern will be compatible with every mapping from the left-hand side of the and. Nevertheless, adding a new triple to the dataset makes $?X$ appear in the evaluation of the right-hand side. Hence, mappings in the evaluation of the left-hand side now share the variable $?X$ with mappings in the evaluation of the right-hand side, so they may be incompatible.

3.3. Well-designed graph patterns and weak-monotonicity

SPARQL graph patterns that do not satisfy the weak-monotonicity condition are the patterns in conflict with the open-world assumption. But as shown above, some non-weakly-monotone graph patterns (like the one presented in Example 4) have a rather unnatural syntactic form: optional information offered to a subpattern can cause an inconsistency with the mappings resulting from the evaluation of other subpatterns. Hence, a natural step towards capturing the set of weakly-monotone SPARQL graph patterns is to disallow this syntactic form. In (Pérez et al., 2009), the authors introduce the concept of well-designedness, a syntactic restriction intended to allow only for weakly-monotone graph patterns.

DEFINITION 3.3.1. *Let P be a SPARQL graph pattern. P is said to be UNION-free well-designed if it belongs to* AOF *and satisfies the following conditions:*

- *for every subpattern of P of the form $(P' \text{ FILTER } R)$, it is the case that $var(R) \subseteq var(P')$, and*
- *for every subpattern of P of the form $(P_1 \text{ OPT } P_2)$, the variables in $var(P_2) \setminus var(P_1)$ are not mentioned in P outside $(P_1 \text{ OPT } P_2)$.*

As shown in (Pérez et al., 2009), this condition indeed allows only for weakly-monotone graph patterns. However, the lack of disjunction makes UNION-free well-designed graph patterns a somewhat restrictive fragment. The concept of well-designedness is extended to include UNION. A graph pattern in AUOF is well-designed if it is of the form

$$P_1 \text{ UNION } P_2 \text{ UNION } \cdots \text{ UNION } P_n,$$

where every disjunct P_i is a UNION-free well-designed graph pattern. These patterns have been widely adopted as a good practice for writing SPARQL queries (see, e.g, (Pichler & Skritek, 2014)). Notice that UNION-free well-designed graph patterns were originally called well-designed graph patterns. By the definition above, in the following well-designed graph patterns are allowed disjunction at the top-most level.

Unfortunately, the definition of well-designed graph patterns does not provide a good insight on their expressive power. In fact, the question of whether every weakly-monotone graph pattern in AUOF is equivalent to a well-designed graph pattern is still open. The first contribution of this dissertation is to show that this is not the case. To formally prove that well-designed graph patterns are not capable of expressing every weakly-monotone graph pattern in AUOF, we use the following two lemmas.

LEMMA 1. *(Pérez et al., 2009) Let G be an RDF graph and let P be a graph pattern in* AOF*. Then, for every two distinct mappings $\mu_1, \mu_2 \in [\![P]\!]_G$, it is the case that $\mu_1 \nsim \mu_2$.*

LEMMA 2. *Let $P = P_1$ UNION P_2 UNION $\cdots$ UNION P_n be a well-designed graph pattern and G_1, G_2 be two RDF graphs such that $G_1 \subseteq G_2$. Then, for every P_i $(1 \leq i \leq n)$ and every two mappings μ_1, μ_2, if $\mu_1 \in [\![P_i]\!]_{G_1}$ and μ_2 is the only mapping in $[\![P]\!]_{G_2}$ subsuming μ_1, then $\mu_2 \in [\![P_i]\!]_{G_2}$.*

PROOF. As P_i is weakly-monotone and $G_1 \subseteq G_2$, there must be a mapping subsuming μ_1 in $[\![P_i]\!]_{G_2}$. Since $[\![P_i]\!]_{G_2} \subseteq [\![P]\!]_{G_2}$, that mapping must be μ_2. □

Now we are ready to prove the aforementioned result. Instead of directly showing that well-designed graph patterns are not capable of expressing every weakly-monotone graph pattern, we provide a stronger result.

THEOREM 1. *There is an* AUOF *weakly-monotone graph pattern that cannot be expressed as a disjunction of* AOF *weakly-monotone graph patterns.*

PROOF. Consider de graph pattern

$$P = (?X, a, b) \text{ OPT } ((?X, c, ?Y) \text{ UNION } (?X, d, ?Z)).$$

It is immediate to show that P is weakly-monotone since both sides of the OPT are monotone. Now we prove that P is not equivalent to any well-designed graph pattern. Define the following RDF graphs

$$\begin{aligned} G_1 &= \{(1, a, b)\} \\ G_2 &= \{(1, a, b), (1, c, 2)\} \\ G_3 &= \{(1, a, b), (1, d, 3)\} \\ G_4 &= \{(1, a, b), (1, c, 2), (1, d, 3)\}. \end{aligned}$$

By semantics of SPARQL we have:

$$\begin{aligned} [\![P]\!]_{G_1} &= \{[?X \to 1]\} \\ [\![P]\!]_{G_2} &= \{[?X \to 1, ?Y \to 2]\} \\ [\![P]\!]_{G_3} &= \{[?X \to 1, ?Z \to 3]\} \\ [\![P]\!]_{G_4} &= \{[?X \to 1, ?Y \to 2], [?X \to 1, ?Z \to 3]\}. \end{aligned}$$

Assume for the sake of contradiction that

$$P \equiv P_1 \text{ UNION } P_2 \text{ UNION } \cdots \text{ UNION } P_n,$$

where every P_i is a weakly-monotone graph pattern in AOF. By semantics of UNION, there must be an i for which $[\![P_i]\!]_{G_1} = \{[?X \to 1]\}$. Without loss of generality we can assume $i = 1$. Now, since $[\![P]\!]_{G_2} = \{[?X \to 1, ?Y \to 2]\}$ and $G_1 \subseteq G_2$, by Lemma 2 we know that $[\![P_1]\!]_{G_2} = \{[?X \to 1, ?Y \to 2]\}$. Moreover, as $[\![P]\!]_{G_3} = \{[?X \to 1, ?Z \to 3]\}$ and $G_1 \subseteq G_3$ it is the case that $[\![P_1]\!]_{G_3} = \{[?X \to 1, ?Z \to 3]\}$.

Since $[?X \to 1, ?Y \to 2] \in [\![P_1]\!]_{G_2}$, $G_2 \subseteq G_4$ and $[\![P]\!]_{G_4} = \{[?X \to 1, ?Y \to 2], [?X \to 1, ?Z \to 3]\}$, by Lemma 2 we have $[?X \to 1, ?Y \to 2] \in [\![P_1]\!]_{G_4}$. The same analysis over G_3 and the mapping $[?X \to 1, ?Z \to 3]$ shows that $[?X \to 1, ?Z \to 3] \in [\![P_1]\!]_{G_4}$.

We conclude that $[\![P_1]\!]_{G_4} = \{[?X \to 1, ?Y \to 2], [?X \to 1, ?Z \to 3]\}$. But since P_1 is in AOF, by Lemma 1 the evaluation of P_1 over G_4 cannot contain two distinct compatible mappings. Thus, we obtain a contradiction and we conclude that P cannot be expressed as a disjunction of weakly-monotone AUOF graph patterns. □

As well-designed graph patterns are weakly-monotone and are in AUOF, we obtain the aforementioned result as a direct corollary.

COROLLARY 1. *The fragment of well-designed graph patterns does not cover the set of all weakly-monotone* AUOF *graph patterns.*

The previous corollary and the proof of Theorem 1 improve our understanding of the expressive power of weakly-monotone and well-designed graph patterns. This negative result motivates the search for new weakly-monotone fragments with more expressive power than that of well-designed graph patterns. To find such fragments, in the next chapter we study techniques developed in first-order logic to capture semantic properties using syntactic conditions. In the subsequent chapters we will show how these techniques can be applied to weak-monotonicity over SPARQL.

4. SYNTAX VERSUS SEMANTICS IN FIRST-ORDER LOGIC

Syntactic characterizations have played a key role in the development of logic, model theory and database theory. For example, the connection between positivity and monotonicity in first-order logic has served for developing several areas of database systems, e.g. consistent query answering (Bertossi, 2006), data exchange (Libkin, 2006), query rewriting (Nash, Segoufin, & Vianu, 2010) and query plans over views (Benedikt, ten Cate, & Tsamoura, 2014). We present an overview of the fundamental techniques used to prove these characterizations, and then recall in detail two results of interest.

We assume the reader is familiar with basic first-order notation and concepts like free variables, predicates, constants, tuples, vocabularies, etc. See (Abiteboul et al., 1995; Enderton & Enderton, 2001) for definitions.

4.1. Interpolation and preservation theorems

During the 1950's, William Craig proved his famous interpolation theorem, in which he established that given two formulas φ and ψ over vocabularies $\mathcal{L}_1$ and $\mathcal{L}_2$, if $\models \varphi \rightarrow \psi$, then there is a formula θ (called an interpolant) over the vocabulary $\mathcal{L}_1 \cap \mathcal{L}_2$, such that $\models \varphi \rightarrow \theta$ and $\models \theta \rightarrow \psi$. This theorem was a real breakthrough in the theoretical study of logics, and immediately proved to have important consequences. Craig showed, for example, that his theorem allows to give a simple proof of Beth's definability theorem (Craig, 1957). It did not took long for the logic community to realize that the relation between the interpolant and the two original formulas played a key role in terms the applications that can be obtained from interpolation. In fact, looking for stronger conditions over the interpolant has been an active area of research from Craig's original prove until today (see (Feferman, 2008) for a good survey, and (Benedikt et al., 2014) for a recent interpolation theorem). In this chapter, we present two interpolation theorems that will be particularly useful for studying weak-monotonicity in SPARQL. The reason because this theorems will be of use is that they allow for giving simple proofs of preservation theorems, which characterize sets of formulas satisfying a semantic property. Preservation theorems receive their name from the fact that the semantic properties involved are defined in terms of *preservation*. For example, the homomorphism preservation theorem states that a formula is preserved under homomorphism (it behaves equally under every two homomorphic structures) if and only if it is logically equivalent to a positive existential formula (a syntactic condition over first-order formulas).

The first interpolation theorem we present in detail was proved by Roger C. Lyndon in 1959. Lyndon generalized Craig's techniques by restricting the *polarity* of the occurrences of predicates in the interpolant. This allows to prove what is known as Lyndon's positivity theorem (also considered one of the preservation theorems), which states that *positive* formulas characterize the set of *monotone* formulas. The second interpolation theorem we are interested in, was proved by Martin Otto (Otto, 2000). This theorem further generalizes Craigs's interpolation theorem by using *relativized* first-order formulas. Otto proved that his theorem unifies a good number of preservation and interpolation theorems, including Lyndon's interpolation. We will show, in particular, how Otto's theorem is applied to give a straightforward proof of the Łoś-Tarski preservation theorem, a classical result in model theory which states that a formula is preserved under *extensions* if and only if it is equivalent to an existential formula.

4.2. Lyndon's interpolation and positive formulas

We start by showing how Lyndon's theorem is used to characterize monotonicity with the set of positive formulas. To formally state this result we need to recall the following definition.

DEFINITION 4.2.1. *Let P be a predicate and φ a first order formula not mentioning* $\{\rightarrow, \leftrightarrow\}$. *An occurrence of P in φ is said to be positive (negative) if it is under an even (odd) number of nested negations. Moreover, φ is said to be positive (negative) in P if every occurrence of P in φ is positive (negative).*

Lyndon's interpolation theorem relates the positive/negative occurrence of predicates in the interpolant with positive/negative occurrence of predicates in the initial formulas (φ and ψ). Its original statement, after modernizing the notation, is as follows:

THEOREM 2. *(Lyndon, 1959) Let φ and ψ be sentences of a vocabulary $\mathcal{L}$ such that $\models \varphi \to \psi$. Then, there is a sentence θ of $\mathcal{L}$ such that $\models \varphi \to \theta$, $\models \theta \to \psi$, and that if a relation symbol occurs positively (negatively) in θ, then it occurs positively (negatively) in both φ and ψ.*

The characterization of monotonicity by means of positivity in first-order logic is one of the most famous and immediate applications of Lyndon's interpolation theorem, and is therefore known as Lyndon's positivity theorem. To formally recall the proof of this characterization, we need the definition of monotonicity for first-order logic:

DEFINITION 4.2.2. *A first-order formula φ over a vocabulary $\mathcal{L}$ is said to be monotone in a predicate P if for every two $\mathcal{L}$-structures $\mathfrak{A}$ and $\mathfrak{B}$, if*

- *every element of $\mathcal{L} \setminus \{P\}$ has the same interpretation in $\mathfrak{A}$ and $\mathfrak{B}$, and*
- *$P^{\mathfrak{A}} \subseteq P^{\mathfrak{B}}$,*

then it is the case that $\mathfrak{A} \models \varphi$ implies $\mathfrak{B} \models \varphi$.

Monotone formulas play an important role in first-order logic, for example they are fundamental to the definition of fixed-point logics (Gurevich & Shelah, 1986; Libkin, 2013; Immerman, 1982; Vardi, 1982). However, the problem of deciding if a first-order formula is monotone is undecidable. Thus, it is natural to look for a decidable syntactic restriction that captures all monotone formulae. Lyndon's theorem precisely allows to find such a characterization.

THEOREM 3 (Lyndon's positivity theorem). *Let $\mathcal{L}$ be a vocabulary and let P a predicate in $\mathcal{L}$. For every $\mathcal{L}$-formula φ, if φ is monotone in P then there is an $\mathcal{L}$-formula ψ that is equivalent to φ and is positive in P.*

PROOF. Let $\mathcal{L}$ be a vocabulary and let P be a predicate in $\mathcal{L}$. Denote by $\mathcal{L}'$ the vocabulary that results from extending $\mathcal{L}$ with a new predicate P' of the same arity as P. Let ϕ be an $\mathcal{L}$-formula that is monotone in P. Consider the following $\mathcal{L}'$-formula:

$$[\phi(P) \wedge P \subseteq P'] \rightarrow \phi(P').$$

This formula explicitly states that if ϕ is satisfied and P' contains more information than P, then ϕ is satisfied when replacing P by P'. It is clear that this is just a reformulation of saying that ϕ is monotone in P, and hence $\models [\phi(P) \wedge P \subseteq P'] \rightarrow \phi(P')$.

Now we apply Lyndon's interpolation theorem. Define the formulas $\varphi = \phi(P) \wedge P \subseteq P'$ and $\psi = \phi(P')$. Since $\models \varphi \rightarrow \psi$, we can deduce the existence of a formula θ such that $\models \varphi \rightarrow \theta$, $\models \theta \rightarrow \psi$ and that every predicate mentioned positively (negatively) in θ is mentioned positively (negatively) in both φ and ψ. Now, since P is not mentioned (either positively or negatively) in ψ, it cannot be mentioned in θ. Hence we denote θ by $\theta(P')$. Moreover, P' can only occur positively in θ as it is only mentioned positively in φ. We conclude that the formula $\theta(P')$ is positive in P'.

We prove now that $\theta(P)$ is equivalent to $\phi(P)$.

- [$\Rightarrow$] Let $\mathfrak{A}$ be an $\mathcal{L}$ structure satisfying $\theta(P)$. Let $\mathfrak{A}'$ be the structure that results from extending $\mathfrak{A}$ to $\mathcal{L}'$ by making $P' = P$. Since P and P' have the same interpretation, $\mathfrak{A}'$ satisfies $\theta(P')$ and hence $\phi(P')$ (as $\models \theta \rightarrow \varphi$). Again, as P and P' have the same interpretation, we have that $\mathfrak{A}'$ satisfies $\phi(P)$. As $\phi(P)$ does not mention P', $\phi(P)$ is also satisfied by $\mathfrak{A}$.
- [$\Leftarrow$] Let $\mathfrak{A}$ be an $\mathcal{L}$ structure satisfying $\phi(P)$. Let $\mathfrak{A}'$ be the structure that results from extending $\mathfrak{A}$ to $\mathcal{L}'$ by making $P' = P$. Since in $\mathfrak{A}'$ it is the case that $P \subseteq P'$, we know that $\mathfrak{A}'$ satisfies $\varphi(P)$. Hence, $\mathfrak{A}'$ satisfies $\theta(P')$ as $\models \varphi \rightarrow \theta$. But since in $\mathfrak{A}'$ the predicates P and P' have the same interpretation, $\mathfrak{A}'$ also satisfies $\theta(P)$. Since $\theta(P)$ does not mention P', we have that $\theta(P)$ is also satisfied in $\mathfrak{A}$.

We have that the original formula $\phi(P)$ is equivalent to $\theta(P)$. This concludes the proof as $\theta(P)$ is positive in P. $\square$

This ends the first application of interpolation theorems to preservation theorems. Next we proceed to show how Otto's interpolation theorem can be applied to prove the Łoś-Tarski preservation theorem.

4.3. Otto's interpolation and the Łoś-Tarski preservation theorem

Otto's interpolation states that if the two original formulas are *relativized* with respect to a set of unary predicates, then so is the interpolant. In this section we show in detail how this result allows to prove the Łoś-Tarski preservation theorem.

DEFINITION 4.3.1. *Let $\mathcal{L}$ be a vocabulary, and let $\mathbb{U}$ be a set of unary predicates in $\mathcal{L}$. An $\mathcal{L}$-formula φ is said to be $\mathbb{U}$-relativized if each quantification in φ is of the form*

$$\exists x(U(x) \wedge \psi(x)) \quad \text{or} \quad \forall x(\neg U(x) \vee \psi(x)),$$

for some $U \in \mathbb{U}$.

Otto's interpolation theorem asserts that, given two $\mathbb{U}$-relativized formulas φ and ψ such that $\models \varphi \to \psi$, there is a $\mathbb{U}$-relativized interpolant satisfying Lyndon's conditions. Moreover, he explicitly includes free variables in his theorem, and proves that the result holds with and without equality. We are particularly interested in the version in which equality is allowed:

THEOREM 4. *(Otto, 2000) Let $\mathcal{L}$ be a vocabulary, and let $\mathbb{U}$ be a set of unary predicates in $\mathcal{L}$. Let φ and ψ be $\mathbb{U}$-relativized formulae such that $\models \varphi \to \psi$. Then there is a formula θ such that*

(1) *θ satisfies the conditions of Lyndon's interpolation theorem w.r.t. φ and ψ,*
(2) *θ is $\mathbb{U}$-relativized,*
(3) *the free variables of θ occur as free variables in both φ and ψ.*

Next, we show how Otto applies his theorem to prove the Łoś-Tarski preservation theorem. We need to introduce the concept of extension for first-order structures.

DEFINITION 4.3.2. *Let $\mathcal{L}$ be a vocabulary and let $\mathfrak{A}$ and $\mathfrak{B}$ be two $\mathcal{L}$-structures. $\mathfrak{B}$ is said to be an extension of $\mathfrak{A}$, denoted by $\mathfrak{A} \subseteq \mathfrak{B}$, if*

- *the domain of $\mathfrak{A}$ is a subset of the domain of $\mathfrak{B}$,*
- *for every predicate $P \in \mathcal{L}$, the relation $P^{\mathfrak{A}}$ contains exactly the tuples in $P^{\mathfrak{B}}$ which only mention elements in the domain of $\mathfrak{A}$,*
- *$c^{\mathfrak{A}} = c^{\mathfrak{B}}$ for every constant $c \in L$*

A formula $\varphi(\bar{x})$ is said to be preserved under extensions if for every tuple $\bar{a}$ and every two structures $\mathfrak{A}$ and $\mathfrak{B}$ such that $\mathfrak{A} \subseteq \mathfrak{B}$, if $\mathfrak{A} \models \varphi(\bar{a})$ then $\mathfrak{B} \models \varphi(\bar{a})$. The Łoś-Tarski preservation theorem states that a formula is equivalent to an existential formula if and only if it is preserved under extensions. Existential formulas are of the form $\exists\bar{x}\varphi$ where φ is free of quantification.

THEOREM 5. *(Łoś-Tarski preservation theorem) Let $\mathcal{L}$ be a vocabulary and let φ be an $\mathcal{L}$-formula. Then, φ is equivalent to an existential formula if and only if it is preserved under extensions.*

PROOF. It is easy to show using induction that an existential formula is preserved under extensions, we only prove the opposite direction. Let $\mathcal{L}'$ be the vocabulary that results from extending $\mathcal{L}$ with two new unary predicates U_1 and U_2. Let $\mathbb{U} = \{U_1, U_2\}$, and let φ be a formula that is preserved under extensions. Define φ^U be the result of *relativising* φ to U, i.e. replacing every existential quantification $\exists x\psi(x)$ by $\exists x(U(x) \wedge \psi(X))$ and every universal quantification $\forall x\psi(x)$ by $\forall x(U(x) \rightarrow \psi(X))$. By using the fact that φ is preserved under extensions, one can prove the next implication to be a tautology:

$$\left[\forall y(U_1(y) \rightarrow U_2(y)) \wedge U_1(\bar{x}) \wedge \varphi^{U_1}(\bar{x})\right] \rightarrow \varphi^{U_2}(\bar{x}).$$

Hence, we can find an $\mathbb{U}$-relativized interpolant with the properties mentioned in Theorem 4. Note that since U_2 is only mentioned positively in the left-hand side, then U_2 can only be mentioned positively in the interpolant. Moreover, U_1 is not mentioned in the right-hand side, so it cannot be mentioned in the interpolant. We conclude the existence of a $\{U_2\}$-relativized formula $\theta^{U_2}(\bar{x})$ in which U_2 is only mentioned positively. Notice that in $\theta^{U_2}(\bar{x})$ no existential quantifier can be under negation, as this would contradict the fact that every occurrence of U_2 is positive. It is a well-known fact that, given a formula φ without universal quantifiers in which all existential quantifiers are mentioned positively, one can construct an equivalent existential formula by transforming φ into Prenex normal form (Abiteboul et al., 1995). Thus, we can assume w.l.o.g. that $\theta^{U_2}(\bar{x})$ is an existential $\{U_2\}$-relativized formula. Let $\theta(\bar{x})$ be the result of replacing every occurrence of U_2 by True in $\theta^{U_2}(\bar{x})$. We have that $\theta(\bar{x})$ is also existential and is not relativized. We show that θ is equivalent to φ.

- [$\Rightarrow$]Let $\mathfrak{A}$ be an $\mathcal{L}$-structure, and let $\bar{a}$ be a tuple such that $\mathfrak{A} \models \varphi(\bar{a})$. Let $\mathfrak{A}'$ be the $\mathcal{L}'$-structure that results from adding to $\mathfrak{A}$ the unary predicates $U_1^{\mathfrak{A}}$ and $U_2^{\mathfrak{A}}$ satisfying $\forall x(U_1(x) \wedge U_2(x))$, and maintaining everything else. It is easy to see that
$$\mathfrak{A}' \models \forall y(U_1(y) \rightarrow U_2(y)) \wedge U_1(\bar{a}) \wedge \varphi^{U_1}(\bar{a})$$
which implies that $\mathfrak{A}' \models \theta^{U_2}(\bar{a})$. But since in $\mathfrak{A}'$ the relation U_2 is the domain of $\mathfrak{A}'$, we know that the relativized and unrelativized versions of the quantifiers are equivalent. Hence, we have $\mathfrak{A}' \models \theta(\bar{a})$. Since θ does not mention U_1 or U_2, we obtain $\mathfrak{A} \models \theta(\bar{a})$, which was to be shown.
- [$\Leftarrow$]Let $\mathfrak{A}$ be an $\mathcal{L}$-structure, and let $\bar{a}$ be a tuple such that $\mathfrak{A} \models \theta(\bar{a})$. Let $\mathfrak{A}'$ be the $\mathcal{L}'$-structure that results from adding to $\mathfrak{A}$ the unary predicates $U_1^{\mathfrak{A}}$ and $U_2^{\mathfrak{A}}$ satisfying $\forall x(U_1(x) \wedge U_2(x))$, and maintaining everything else. It is easy to see that $\mathfrak{A}' \models \theta^{U_2}(\bar{a})$, which implies that $\mathfrak{A}' \models \varphi^{U_2}(\bar{a})$. But since in $\mathfrak{A}'$ the relation U_2 has every element, we know that the relativized and unrelativized versions of the

quantifiers are equivalent. Hence, we have $\mathfrak{A}' \models \varphi(\bar{a})$. Since φ does not mention U_1 or U_2, we obtain $\mathfrak{A} \models \varphi(\bar{a})$, which was to be shown.

We have that $\varphi(\bar{x})$ is equivalent to $\theta(\bar{x})$, completing the proof. □

This concludes the second application of interpolation to preservation theorems that will be used in the next chapter.

Before adapting the presented techniques to weak-monotonicity and SPARQL, we need to discuss the validity of preservation theorems when we restrict only to finite models, as RDF graphs are defined as finite sets of triples.

4.4. Preservation theorems and finite models

The two interpolation theorems presented above were originally proved assuming no restriction over the first-order structures. In particular, they assume that (the domain of) structures can be either finite or infinite. Although assuming arbitrary structures might seem more general, for the interpolation theorems it turns out to be a restriction: Given two formulas φ and ψ, it can be the case that $\not\models \varphi \rightarrow \psi$, but every finite structure satisfying φ also satisfies ψ. If this is the case then we say that $\varphi \rightarrow \psi$ is a tautology under finite models, and write $\models_{\text{FIN}} \varphi \rightarrow \psi$. Notice that if $\models \varphi \rightarrow \psi$ then we immediately have $\models_{\text{FIN}} \varphi \rightarrow \psi$. However, it can be the case that $\models_{\text{FIN}} \varphi \rightarrow \psi$, but $\mathfrak{B} \not\models \varphi \rightarrow \psi$ for some infinite structure $\mathfrak{B}$. In particular, this has its consequence over monotonicity and preservation under extensions. A formula could be monotone or preserved under extensions over finite models, but not when considering arbitrary models. The interpolation theorems provide the existence of an interpolant with the desired properties only when $\varphi \rightarrow \psi$ is a tautology over arbitrary structures, but this does not not imply the existence of an interpolant when $\models_{\text{FIN}} \varphi \rightarrow \psi$.

The study of properties over finite models might seem just a mathematical problem, but it is of particular interest in the formal study of databases, essentially because databases are finite collections. It is interesting to mention that the study of finite model theory has widely diverged from the study of classical model theory (Libkin, 2013). The reason relies on the fact that some of the fundamental theoretical tools over arbitrary models fail when restricting to finite models. The most notable example is probably the compactness theorem proved Kurt Gödel in 1929, which fails for finite models. To the best of our knowledge, the compactness theorem is used in most proofs of interpolation theorems. Given that some classical techniques fail in the finite case, it is natural to ask whether the interpolation and preservation theorems hold in the finite.

The homomorphism preservation theorem is considered to be an exception, as it was recently proved to hold in the finite case (Rossman, 2008). Unfortunately, the same cannot be affirmed for the two results that will be applied to weak-monotonicity. The failure of the Łoś-Tarski theorem in the finite case (Tait, 1959) immediately implies that Otto's interpolation theorem does not hold in the finite case either, as otherwise the presented proof of the Łoś-Tarski theorem would also hold under finite models. It is also known that Lyndon's positivity theorem fails when considering only finite models (Ajtai & Gurevich, 1987), which implies that Lyndon's interpolation theorem also fails in this case. However, it is well-known that the counterexamples to show that the results fail in the finite case are hard to construct, and hence it is not expected to find them in real-world applications.

The fact that positivity does not characterize monotonicity over first-order logic in the finite case does not imply that it is not possible to find such characterization. However, finding a characterization for monotonicity over finite models has proven to be a hard problem, and is in fact still open. We aim to apply interpolation to the case of weak-monotonicity in SPARQL over infinite models. Therefore, in the next section we start by introducing the concept of infinite RDF graphs, which will allow us to present a framework for applying interpolation techniques to SPARQL.

5. INTERPOLATION APPLIED TO WEAK-MONOTONICITY

In the previous chapter, we presented in detail two applications of interpolation theorems to the characterization of semantic properties. Next, we present a framework that allows us to extend these results and find such characterizations in SPARQL. In this framework, we establish a connection between SPARQL and FO using ideas similar to those presented in the translation from SPARQL to non-recursive Datalog with safe negation developed in (Angles & Gutierrez, 2008b) and (Polleres & Wallner, 2013).

As already discussed, to correctly apply interpolation to SPARQL we need to work over arbitrary first order models. Hence, we extend the definition of RDF graphs by allowing every nonempty subset of $\mathbf{I} \times \mathbf{I} \times \mathbf{I}$. It easy to see that the definition of the syntax and semantics presented in Chapter 2 are not affected by this extension, nor are the definition of properties like weak-monotonicity and monotonicity. Notice, however, that a graph pattern that is (weakly-)monotone under the original definition of RDF, namely finite graphs, may no longer have this property under arbitrary models.

5.1. From SPARQL to First-Order Logic

The application of Lyndon's and Otto's interpolation theorems to prove preservation theorems can be divided into three fundamental steps. First, create a formula defining the corresponding semantic property, over which the interpolation theorem can be applied. Second, prove a specific relation between the interpolant and the original formula. Third, show that the interpolant belongs to some syntactic fragment of interest. In this section, we present a translation from graph patterns to first-order formulas that will allow us to perform these three steps over SPARQL graph patterns. Moreover, this translation is simpler than those presented before, and therefore it is also a contribution of this dissertation. To apply interpolation and translate the obtained results to SPARQL, we will also need a transformation from first-order logic to SPARQL, which will be presented in the next section. We now start establishing the transformation from RDF and SPARQL to first-order structures and formulas.

Given a fixed set $I \subseteq \mathbf{I}$ of IRIs, define $\mathcal{L}^{I}_{\text{RDF}}$ as the first-order vocabulary containing a ternary predicate T, a unary predicate Dom, a distinct constant c_i for each $i \in I$, and a new constant n. We first define a correspondence between RDF graphs and first-order structures.

Definition 5.1.1. *Let G be an RDF graph, and let $\mathbf{I}(G)$ be the set of IRIs occurring in the triples of G. We say that an $\mathcal{L}^{\mathbf{I}(G)}_{RDF}$-structure $\mathfrak{A} = \langle D, T^{\mathfrak{A}}, Dom^{\mathfrak{A}}, \{c_i^{\mathfrak{A}}\}_{i \in I}, n\rangle$ corresponds to G if:*

- *for every $i \in \mathbf{I}(G)$ it is the case that $c_i^{\mathfrak{A}} = i$,*
- *G is the set of triples in $T^{\mathfrak{A}} \cap (Dom^{\mathfrak{A}} \times Dom^{\mathfrak{A}} \times Dom^{\mathfrak{A}})$,*
- *There is an element $N \in D$ such that $n^{\mathfrak{A}} = N$.*
- *$N \notin Dom^{\mathfrak{A}}$ and N does not occur in $T^{\mathfrak{A}}$.*

Notice that the correspondence above is not one-to-one in the sense that two structures can correspond to the same RDF graph G. This is a consequence of the introduction of the predicate Dom, which will be fundamental to define relativized quantifiers when applying Otto's interpolation.

Next, we need to state a correspondence between mappings and first-order tuples. To this end, we assume a total strict order $<$ over all variables in $\mathbf{V}$. Given a finite subset X of $\mathbf{V}$, denote by $\bar{X}$ the tuple of variables in X ordered under $<$.

Definition 5.1.2. *Let μ be a mapping and let X a finite subset of $dom(\mu)$. Then, $\mu[\bar{X}]$ is the tuple that results from replacing in $\bar{X}$ every component in $X \cap dom(\mu)$ by its image under μ, and removing every other component.*

To transform a SPARQL graph pattern P into a first-order formula, we take an intermediate step: we create one formula for each subset of var(P). Intuitively, the formula corresponding to a subset X of var(P) will *output* the tuples corresponding to mappings that bind exactly X. We abuse notation by not distinguishing between FO and SPARQL variables. We also abuse notation by assuming $\mu(i) = i$ for every mapping μ and $i \in \mathbf{I}$.

LEMMA 3. *For every graph pattern P, there is a set $\{\varphi^P_X(\bar{X})\}_{X \subseteq var(P)}$ of formulas in $\mathcal{L}^{\mathbf{I}(P)}_{RDF}$ such that, for every mapping μ and RDF graph G, it is the case that $\mu \in [\![P]\!]_G$ if and only if $\mathfrak{A} \models \varphi^P_{dom(\mu)}(\mu[\bar{X}])$ for every $\mathcal{L}^{\mathbf{I}(P)}_{RDF}$-structure $\mathfrak{A}$ corresponding to G.*

PROOF. Let P be a SPARQL pattern. We proceed by induction on the structure of P.

- Let $P = (t_1, t_2, t_3)$ be a triple pattern and let $\mathfrak{A}$ be an $\mathcal{L}^{\mathbf{I}(P)}_{\text{RDF}}$-structure corresponding to G. Since for every RDF graph G and mapping $\mu \in [\![P]\!]_G$ we have var(P) = dom(μ), define $\varphi^P_X(\bar{X})$ as a contradiction for every $X \subsetneq$ var(P). For X = var(P) define

$$\varphi^P_X(\bar{X}) = T(t_1, t_2, t_3) \wedge Dom(t_1) \wedge Dom(t_2) \wedge Dom(t_3).$$

  A mapping μ belongs to $[\![P]\!]_G$ if and only if $(\mu(t_1), \mu(t_2), \mu(t_3))$ belongs to G. But this occurs if and only if $\mu(t_1), \mu(t_2), \mu(t_3) \in Dom^{\mathfrak{A}}$ and $(\mu(t_1), \mu(t_2), \mu(t_3)) \in T^{\mathfrak{A}}$.
- Let $P = P_1$ UNION P_2 and let $\mathfrak{A}$ be an $\mathcal{L}^{\mathbf{I}(P)}_{\text{RDF}}$-structure corresponding to G. For every $X \subseteq$ var(P) define $\varphi^P_X(\bar{X})$ as

$$\varphi^P_X(\bar{X}) = \varphi^{P_1}_X(\bar{X}) \vee \varphi^{P_2}_X(\bar{X}).$$

  Let μ be a mapping and X = dom(μ). By semantics of UNION, $\mu \in [\![P]\!]_G$ if and only if $\mu \in [\![P_1]\!]_G \cup [\![P_2]\!]_G$. Hence, by hypothesis we have $\mu \in [\![P]\!]_G$ if and only if $\mathfrak{A} \models \varphi^{P_1}_X(\mu[\bar{X}])$ or $\mathfrak{A} \models \varphi^{P_2}_X(\mu[\bar{X}])$, which is the semantic definition of $\mathfrak{A} \models \varphi^{P_1}_X(\mu[\bar{X}]) \vee \varphi^{P_2}_X(\mu[\bar{X}])$.
- Let $P = P_1$ AND P_2 and let $\mathfrak{A}$ be an $\mathcal{L}^{\mathbf{I}(P)}_{\text{RDF}}$-structure corresponding to G. For every $X \subseteq$ var(P) define the formula $\varphi^P_X(\bar{X})$ as

$$\varphi^P_X(\bar{X}) = \bigvee_{X_1 \cup X_2 = X} \left[\varphi^{P_1}_{X_1}(\bar{X}_1) \wedge \varphi^{P_2}_{X_2}(\bar{X}_2)\right].$$

Let G and μ be an RDF graph and a mapping, respectively, and let $X = \text{dom}(\mu)$. If μ belongs to $[\![P]\!]_G$, then there are two compatible mappings $\mu_1 \in [\![P_1]\!]_G$ and $\mu_2 \in [\![P_2]\!]_G$ such that $\mu = \mu_1 \cup \mu_2$. Let $X_1 = \text{dom}(\mu_1)$ and $X_2 = \text{dom}(\mu_2)$. By hypothesis, we know that $\mathfrak{A} \models \varphi^{P_1}_{X_1}(\mu_1[\bar{X}_1])$ and $\mathfrak{A} \models \varphi^{P_2}_{X_2}(\mu_2[\bar{X}_2])$ which is equivalent to $\mathfrak{A} \models \varphi^{P_1}_{X_1}(\mu_1[\bar{X}_1]) \wedge \varphi^{P_2}_{X_2}(\mu_2[\bar{X}_2])$. As $X_1 \cup X_2 = X$ we have $\mathfrak{A} \models \varphi^{P}_{X}(\mu[\bar{X}])$.

For the converse, if $\mathfrak{A} \models \varphi^{P}_{\text{dom}(\mu)}(\mu[\bar{X}])$ then there are two sets X_1 and X_2 such that $X_1 \cup X_2 = X$ and both $\mathfrak{A} \models \varphi^{P_1}_{X_1}(\mu(\bar{X}_1))$ and $\mathfrak{A} \models \varphi^{P_1}_{X_2}(\mu(\bar{X}_2))$ hold. Define μ_i as μ restricted to X_i ($i \in \{1, 2\}$). It follows from the hypothesis that $\mu_1 \in [\![P_1]\!]_G$ and $\mu_2 \in [\![P_2]\!]_G$. Since μ_1 and μ_2 are compatible (they are both restrictions of μ) and $\mu = \mu_1 \cup \mu_2$, this implies $\mu \in [\![P]\!]_G$.

- Let $P = P_1$ OPT P_2 and let $\mathfrak{A}$ be an $\mathcal{L}^{\mathbf{I}(P)}_{\text{RDF}}$-structure corresponding to G. For every $X \subseteq \text{var}(P)$ define the formula $\varphi^{P}_{X}(\bar{X})$ as

$$\varphi^{P}_{X}(\bar{X}) = \varphi^{P_1 \text{ AND } P_2}_{X}(\bar{X}) \vee \varphi^{P}_{\mathsf{MINUS}\ X}(\bar{X})$$

where $\varphi^{P}_{\mathsf{MINUS}\ X}(\bar{X})$ is defined as

$$\varphi^{P_1}_{X}(\bar{X}) \wedge \neg \bigvee_{X' \subseteq \text{var}(P_2)} \exists (X' \setminus X) \left(\bigwedge_{x' \in X'} Dom(x') \wedge \varphi^{P_2}_{X'}(\bar{X}') \right).$$

Notice that in the definition of the MINUS formula, the mention of Dom is not necessary because elements coming from $\varphi^{P_2}_{X'}(\bar{X}')$ are already binded to $Dom^{\mathfrak{A}}$. However, this mention allows us to directly conclude that a graph pattern is transformed into a $\{Dom\}$-relativized formula. Now we proceed with the proof.

Let G and μ be an RDF graph and a mapping, respectively, and let $X = \text{dom}(\mu)$. By definition, μ belongs to $[\![P_1 \text{ AND } P_2]\!]_G$ or to $[\![P_1]\!]_G \setminus [\![P_2]\!]_G$. In the first case, we know that $\mathfrak{A} \models \varphi^{P_1 \text{ AND } P_2}_{X}(\mu[\bar{X}])$. It remains show that if $\mu \in [\![P_1]\!]_G \setminus [\![P_2]\!]_G$ then $\mathfrak{A} \models \varphi^{P}_{\mathsf{MINUS}\ ,X}(\mu[\bar{X}])$. As $\mu \in [\![P_1]\!]_G$, we know $\mathfrak{A} \models \varphi^{P_1}_{X}(\mu[\bar{X}])$, so we only need to prove that there is no set $X' \subseteq \text{var}(P_2)$ such that $\mathfrak{A} \models \varphi^{P_2}_{X'}(\mu'[\bar{X}'])$ for some

μ' compatible with μ. But if this was the case then μ' would be in $[\![P_2]\!]_G$, which contradicts the fact that $\mu \in [\![P_1]\!]_G \setminus [\![P_2]\!]_G$ (since μ' and μ are compatible).

For the converse, assume $\mathfrak{A} \models \varphi^P_X(\mu[\bar{X}])$ where $X = \text{dom}(\mu)$. If it is the case that $\mathfrak{A} \models \varphi^{P_1 \text{ AND } P_2}_X(\mu[\bar{X}])$, we know by the AND case that $\mu \in [\![P_1 \text{ AND } P_2]\!]_G$ and hence $\mu \in [\![P]\!]_G$. The remaining case is when $\mathfrak{A} \models \varphi^P_{\text{MINUS } X}(\mu[\bar{X}])$. If this is the case, by hypothesis it readily follows that $\mu \in [\![P_1]\!]_G$. Now we have to prove that μ is not compatible with any mapping in $[\![P_2]\!]_G$. Proceed by contradiction. Assume there is a mapping $\mu' \in [\![P_2]\!]_G$ compatible with μ. We know $\mathfrak{A} \models \varphi^{P_2}_{X'}(\mu'[\bar{X}'])$ where $X' = \text{dom}(\mu')$. Since μ and μ' are compatible, μ' can be obtained by extending the assignments in μ, and thus $\mathfrak{A}$ would not satisfy $\varphi^P_{\text{MINUS } X}(\mu[\bar{X}])$, which leads to a contradiction.

- Let $P = \text{SELECT } V \text{ WHERE } Q$ and let $\mathfrak{A}$ be an $\mathcal{L}^{\mathbf{I}(P)}_{\text{RDF}}$-structure corresponding to G. For every $X \subseteq \text{var}(P)$ such that $V \not\subseteq X$, the formula $\varphi^P_X(\bar{X})$ is defined as a contradiction. It is immediate to show that this satisfies the equivalence, as P cannot output variables not mentioned in V. Now, for every $X \subseteq \text{var}(P) \cap V$ define the formula $\varphi^P_X(\bar{X})$ as

$$\varphi^P_X(\bar{X}) = \bigvee_{Y \subseteq \text{var}(P) | X \subseteq Y} \exists (Y \setminus X) \left(\bigwedge_{y \in Y} Dom(y) \wedge \varphi^Q_Y(\bar{Y}) \right)$$

As in the OPT case, including Dom is not necessary but allows us to state that the resulting formula is $\{Dom\}$-relativized. Let G and μ be an RDF graph and a mapping, respectively, and let $X = \text{dom}(\mu)$. If μ belongs to $[\![P]\!]_G$, then there is a mapping $\mu' \in [\![Q]\!]_G$ such that $\mu'_{|X} = \mu$. Let $Y = \text{dom}(\mu')$. We have by hypothesis that $\mathfrak{A} \models \varphi^Q_Y(\mu'[\bar{Y}])$. Since $\mu'[\bar{Y}]$ is a tuple extending $\mu[\bar{X}]$, we have that $\mathfrak{A} \models \exists (Y \setminus X) \varphi^Q_Y(\mu'[\bar{Y}])$ when replacing the free variables in X according to $\mu[\bar{X}]$. It readily follows that $\mathfrak{A} \models \varphi^P_X(\mu[\bar{X}])$.

For the converse, if $\mathfrak{A} \models \varphi^P_{\text{dom}(\mu)}(\mu[\bar{X}])$, there must be a tuple $\bar{a}$ that extends $\mu[\bar{X}]$ and a set of variables Y with $X \subseteq Y \subseteq \text{var}(P)$, such that $\mathfrak{A} \models \varphi^Q_Y(\bar{a})$. Let μ' be the tuple corresponding to $\bar{a}$ (i.e. $\mu'^{Y}_{\text{FO}} = \bar{a}$). By hypothesis, we have that $\mu' \in [\![Q]\!]_G$.

But since μ' corresponds to $\bar{a}$, and $\bar{a}$ extends $\mu[\bar{X}]$, we have that $\mu'_{|X} = \mu$. We conclude that $\mu \in [\![\text{SELECT } V \text{ WHERE } Q]\!]_G$.

- Let $P = P_1$ FILTER R and let $\mathfrak{A}$ be an $\mathcal{L}^{\mathbf{I}(P)}_{\text{RDF}}$-structure corresponding to G. For every $X \subseteq \text{var}(P)$ define $\varphi^P_X(\bar{X})$ as

$$\varphi^P_X(\bar{X}) = \varphi^{P_1}_X(\bar{X}) \wedge \varphi_R(\bar{X})$$

where $\varphi_R(\bar{X})$ is inductively defined as follows:

  - If R is an equality and $\text{var}(R) \nsubseteq X$, then $\varphi_R = \text{False}$.
  - If R is an equality and $\text{var}(R) \subseteq X$, then $\varphi_R = R$.
  - If $R = \mathit{bound}(x)$ and $x \notin X$ then $\varphi_R = \text{False}$.
  - If $R = \mathit{bound}(x)$ and $x \in X$ then $\varphi_R = \mathit{True}$.
  - If R is of the form $\neg R_1$, $R_1 \wedge R_2$, or $R_1 \vee R_2$ for filter conditions R_1 and R_2, then φ_R is the corresponding boolean combination of φ_{R_1} and φ_{R_2}.

Let D and μ be an RDF dataset and a mapping, respectively, and let $X = \text{dom}(\mu)$. It is easy to see from the definition of φ_R that $\mathfrak{A} \models \varphi_R(\mu[\bar{X}])$ if and only if $\mu \models R$. By hypothesis we have $\mu \in [\![P_1]\!]_G$ if and only if $\mathfrak{A} \models \varphi^{P_1}_{\text{dom}(\mu)}(\mu[\bar{X}])$, and hence it readily follows that $\mathfrak{A} \models \varphi^{P_1}_{\text{dom}(\mu)}(\mu[\bar{X}]) \wedge \varphi_R(\bar{X})$ if and only if $\mu \in [\![P_1]\!]_G$ and $\mu \models R$, which was to be shown.

□

The previous Lemma allows us to take a graph pattern P and construct a set of formulae that together, in a sense, are equivalent to P. We now need to transform such set into one single formula. The main issue in this transformation is that variables might not be binded by mappings, producing a problem in the set of free variables. Here is where we need the constant n and the element N in the domain of structures.

DEFINITION 5.1.3. *Let μ be a mapping and X a finite subset of* $\mathbf{V}$. *Assume $X = \{x_1, \ldots, x_n\}$ and that $i < j$ implies $x_i < x_j$. The tuple representing μ over X, denoted by μ^X_{FO}, is defined as $(a_i)^n_{i=1}$ where $a_i = \mu(x_i)$ if $x_i \in dom(\mu)$ and $a_i = N$ otherwise.*

This definition allows us to output tuples corresponding to mappings with different domains from a single FO formula. We abuse notation and write μ^P_{FO} instead of $\mu^{\mathrm{var}(P)}_{\mathrm{FO}}$, as we are particularly interested in those cases when the set of variables comes from a graph pattern. For the sake of simplicity, we introduce the notion of equivalence between SPARQL graph patterns and first-order formulas:

DEFINITION 5.1.4. *Let P be a graph pattern and let φ be an $\mathcal{L}^{\mathbf{I}(P)}_{RDF}$-formula. We say that P and φ are equivalent, denoted by $P \equiv \varphi$, if for every RDF graph G and mapping μ, it is the case that $\mu \in [\![P]\!]_G$ if and only if $\mathfrak{A} \models \varphi(\mu^P_{FO})$ for every $\mathcal{L}^{\mathbf{I}(P)}_{RDF}$-structure $\mathfrak{A}$ corresponding to G.*

Now we are ready to formally state the aforementioned transformation from SPARQL to FO.

THEOREM 6. *Let P be a SPARQL graph pattern. There is a first-order formula φ_P in $\mathcal{L}^{\mathbf{I}(P)}_{RDF}$ which is equivalent to P.*

PROOF. Let P be a SPARQL graph pattern. Let $\{\varphi^P_X(\bar{X})\}_{X \subseteq \mathrm{var}(P)}$ be the formulas obtained from applying Lemma 3 to P. Now, define the first-order formula $\varphi_P(\mathrm{var}(P))$ as follows:

$$\varphi_P(\mathrm{var}(P)) = \bigvee_{X \subseteq \mathrm{var}(P)} \left[\varphi^P_X(\bar{X}) \wedge \bigwedge_{y \in \mathrm{var}(P) \setminus X} y = n \right]$$

We show that $P \equiv \varphi_P$. Let $\mathfrak{A}$ be an $\mathcal{L}^{\mathbf{I}(P)}_{\mathrm{RDF}}$-structure corresponding to G.

- [$\Rightarrow$] Let $\mu \in [\![P]\!]_G$ and let $X = \mathrm{dom}(\mu)$. Since μ^P_{FO} is a tuple that extends $\bar{X}$ by making variables in $\mathrm{var}(P) \setminus X$ equal to n, we have $\mathfrak{A} \models y = n$ for every $y \in \mathrm{var}(P) \setminus X$ when evaluated over μ^P_{FO}. Also, we know from Lemma 3 that $\mathfrak{A} \models \varphi^P_X(\mu[\bar{X}])$. Hence, we have that $\mathfrak{A} \models \varphi_P(\mu[\bar{X}])$, which was to be shown.
- [$\Leftarrow$] Let μ be a mapping such that $\mathfrak{A} \models \varphi_p(\mu^P_{\mathrm{FO}})$. Since the variables equal to n in μ^P_{FO} are precisely those not in $\mathrm{dom}(\mu)$, the only disjunct that can be satisfied

is that in which $X = \mathrm{dom}(\mu)$. Hence, we know that $\mathfrak{A} \models \varphi^P_{\mathrm{dom}(\mu)}(\mu[\bar{X}])$. From Lemma 3 this directly implies $\mu \in [\![P]\!]_G$, concluding the proof.

□

Notice that the previous transformation creates an FO formula with the same set of free variables that the original graph pattern.

Having a transformation from SPARQL to FO, we can now apply the interpolation techniques explained in Section 4.1. Notice that the formula resulting from a graph pattern might have equalities, and hence we use the version of Otto's interpolation theorem in which equalities are allowed. It is worth mentioning that Lyndon's interpolation theorem can be extended to allow for equalities, but Lyndon did not explicitly included them in the original statement. The first extension to Lyndon's interpolation theorem that included equalities[1] was presented in (Oberschelp, 1968).

To apply interpolation we need to write a formula expressing weak-monotonicity in our FO setting. Given two tuples $\bar{x} = (x_1, \ldots, x_n)$ and $\bar{y} = (y_1, \ldots, y_n)$ of the same length, define the formula $\bar{x} \preceq \bar{y}$ by

$$\bar{x} \preceq \bar{y} = \bigwedge_{i=1}^{n} (x_i = y_i \vee x_i = n).$$

It can be seen that given two mappings μ_1 and μ_2, it is the case that μ_1 is subsumed by μ_2 if and only if ${\mu_1}^V_{\mathrm{FO}} \preceq {\mu_2}^V_{\mathrm{FO}}$ for every set V of variables. We use the relation $\preceq$ to say that a first order formula is weakly-monotone. We also use, for every $I \subseteq \mathbf{I}$, the extended vocabulary ${\mathcal{L}^I_{\mathrm{RDF}}}' = \mathcal{L}_{\mathrm{RDF}} \cup \{T', Dom'\}$, where T' is a new ternary predicate and Dom' is a new unary predicate. Let $I \subseteq \mathbf{I}$ and let φ be an $\mathcal{L}^I_{\mathrm{RDF}}$-formula. The next ${\mathcal{L}^I_{\mathrm{RDF}}}'$-formula asserts that φ is weakly-monotone over structures that correspond to RDF graphs.

$$[\varphi(T, Dom, \bar{x}) \wedge T \subseteq T' \wedge Dom \subseteq Dom'] \rightarrow \exists \bar{y}(\bar{x} \preceq \bar{y} \wedge \varphi(T', Dom', \bar{y})). \tag{5.1}$$

[1]The interpolation property over the polarity of predicates does not hold for equalities. However, there is a weaker condition for the interpolant over the occurrences of equalities and inequalities.

However, if φ corresponds to a weakly-monotone graph pattern, the previous formula is not necessarily a tautology. This occurs because there are structures that do not correspond to any RDF graph, and we do not know how φ will behave over those structures. To actually create a tautology, define the following $\mathcal{L}^{I}_{\text{RDF}}$-sentence:

$$\Sigma_{\text{RDF}} = \bigwedge_{i \in I} c_i \neq n \wedge \bigwedge_{i \neq j} c_i \neq c_j \wedge \neg Dom(n).$$

It is easy to see that an $\mathcal{L}^{I}_{\text{RDF}}$-structure satisfies Σ_{RDF} if and only if it corresponds to some RDF graph. Hence, the following formula states that φ is weakly-monotone over structures that correspond to RDF graphs.

$$[\Sigma_{\text{RDF}} \wedge \varphi(T, Dom, \bar{x}) \wedge T \subseteq T' \wedge Dom \subseteq Dom'] \rightarrow \exists \bar{y}(\bar{x} \preceq \bar{y} \wedge \varphi(T', Dom', \bar{y})). \quad (5.2)$$

Assume now that we have a weakly-monotone graph pattern P, and let φ be the $\mathcal{L}^{\mathbf{I}(P)}_{\text{RDF}}$-formula obtained from applying Theorem 6 to P. Since φ is equivalent to P (which is weakly-monotone), we know that the above implication is a tautology. Therefore, we can obtain an interpolant $\theta(T', Dom', \bar{x})$ satisfying the conditions from Ottos's interpolation theorem. Unfortunately, and as opposed to the applications of interpolation in FO, this interpolant is not necessarily equivalent to φ. In particular, we will see that the interpolant it is not only weakly-monotone, but also monotone. However, there is a strong connection between φ and θ, which is formalized by the following definition.

DEFINITION 5.1.5. *Let P be a graph pattern and let φ be an $\mathcal{L}_{RDF}$-formula. P is said to be equivalent in maximal answers to φ, denoted by $P \equiv_{MAX} \varphi$ if for every RDF graph G and mapping μ, it is the case that μ is a maximal mapping (w.r.t. $\preceq$) in $[\![P]\!]_G$ if and only if μ^P_{FO} is a maximal tuple (w.r.t. $\preceq$ in FO) such that $\mathfrak{A} \models \varphi(\mu^P_{FO})$ for every $\mathcal{L}_{RDF}$-structure corresponding to G.*

Now we are ready to prove the main result obtained from translating SPARQL to FO and applying interpolation.

THEOREM 7. *Let P be a weakly-monotone graph pattern. There is an existential $\mathcal{L}_{RDF}^{\mathbf{I}(P)}$-formula ψ that is positive in both T and Dom, and such that $P \equiv_{MAX} \psi$.*

PROOF. Let P be a weakly-monotone graph pattern and let φ be the $\mathcal{L}_{\text{RDF}}^{\mathbf{I}(P)}$-formula obtained from Theorem 6. Since φ is equivalent to P, which is weakly-monotone, we know that formula 5.2 is a tautology. Then, there is an interpolant θ that satisfies the next conditions:

(1) $[\Sigma_{\text{RDF}} \wedge \varphi(T, Dom, \bar{x}) \wedge T \subseteq T' \wedge Dom \subseteq Dom'] \rightarrow \theta(T', Dom', \bar{x})$,
(2) $\theta(T', Dom', \bar{x}) \rightarrow \exists \bar{y}(\bar{x} \preceq \bar{y} \wedge \varphi(T', Dom', \bar{y}))$,
(3) θ only mentions the predicates T' and Dom',
(4) θ is positive in both T' and Dom'.

As φ and Σ_{RDF} are $\{Dom, Dom'\}$-relativized and Dom is not mentioned in θ, we can deduce that θ is $\{Dom'\}$-relativized. Moreover, Dom' is mentioned positively in θ from which we conclude, as in the first-order case, that θ is an existential formula. Now it only remains to show that φ and θ coincide in their maximal answers. Let $\mathfrak{A}$ be an $\mathcal{L}_{\text{RDF}}^{\mathbf{I}(P)}$-structure corresponding to an RDF graph G. Define $\mathfrak{A}'$ as the structure in $\mathcal{L}_{\text{RDF}}^{\mathbf{I}(P)\,\prime}$ that results from extending $\mathfrak{A}$ with $T' = T$ and $Dom' = Dom$.

- [$\Rightarrow$] For this direction we prove a stronger result: every answer to $\varphi(T, Dom)$ is also an answer to $\theta(T, Dom)$. Let $\bar{a}$ be a tuple such that $\mathfrak{A} \models \varphi(T, Dom, \bar{a})$. Since $\varphi(T, Dom, \bar{x})$ does not mention T' nor Dom', we know that $\mathfrak{A}' \models \varphi(T, Dom, \bar{a})$. We also know that both $T \subseteq T'$ and $Dom \subseteq Dom'$ hold in $\mathfrak{A}$, so by (1) we have that $\mathfrak{A}' \models \theta(T', Dom', \bar{a})$. It immediately follows that $\mathfrak{A} \models \theta(T, Dom, \bar{a})$.
- [$\Leftarrow$] Let $\bar{a}$ be a maximal tuple such that $\mathfrak{A} \models \theta(T, Dom, \bar{a})$. Since $\theta(T, Dom, \bar{x})$ does not mention T' not Dom', and in $\mathfrak{A}'$ we have $T = T'$ and $Dom = Dom'$, we know that $\mathfrak{A}' \models \theta(T', Dom', \bar{a})$. By (2), there must be a tuple $\bar{b}$ subsuming $\bar{a}$ such that $\mathfrak{A}' \models \varphi(T', Dom', \bar{b})$. From [$\Rightarrow$], we know that this implies $\mathfrak{A}' \models \theta(T', Dom', \bar{b})$. Since $\bar{a}$ is a maximal tuple satisfying $\mathfrak{A}' \models \theta(T', Dom', \bar{x})$, we

have that $\bar{a} = \bar{b}$. We conclude that that $\mathfrak{A}' \models \varphi(T', Dom', \bar{a})$. It immediately follows that $\mathfrak{A} \models \varphi(T, Dom, \bar{a})$.

We have that θ is a positive existential formula that is equivalent in maximal answers to P, concluding the proof. $\square$

The previous theorem establishes what we obtain from applying interpolation to formulas related to weakly-monotone SPARQL graph patterns. We now know that given a weakly-monotone SPARQL graph pattern P, there is a positive existential formula φ that is equivalent in maximal answers to P.

5.2. From existential positive FO to SPARQL

We have proved that given a graph pattern P in SPARQL, there is an existential positive formula φ such that $P \equiv_{\mathrm{MAX}} \varphi$. In this section, we discuss how to transform such formula φ back into SPARQL, and what is the syntactic form of the obtained graph pattern. In this transformation we do not need the equivalence for all structures that correspond to RDF graphs. Instead, we use a weaker notion of equivalence in which the correspondence between the formula and the graph pattern only holds in very specific structures.

DEFINITION 5.2.1. *Let G be an RDF graph. The first-order structure that represents G is denoted by G_{FO}, and is defined as the only $\mathcal{L}_{RDF}^{\mathbf{I}(G)}$-structure with domain $\mathbf{I}(G) \cup \{N\}$ that corresponds to G.*

Notice that in a structure representing a graph G, *Dom* is interpreted as $\mathbf{I}(G)$ and T corresponds precisely to the triples in G. Now we define the notion of equivalence that will hold when transforming a first-order positive existential formula into a SPARQL graph pattern.

DEFINITION 5.2.2. *Given an $\mathcal{L}_{RDF}^{\mathbf{I}(P)}$-formula φ and a graph pattern P, we say that P and φ are equivalent in RDF structures, denoted by $P \equiv_{RDF} \varphi$, if for every mapping μ and RDF graph G, it is the case that $\mu \in [\![P]\!]_G$ if and only if $G_{FO} \models \varphi(\mu_{FO}^{P})$.*

We also use the relation $\equiv_{\text{RDF}}$ between two FO formulas to assert that they coincide in every structure representing an RDF graph. Now we proceed to define the aforementioned transformation: given a positive existential formula φ, we construct a graph pattern P such that $\varphi \equiv_{\text{RDF}} P$. To this end, we first transform our formula into a union of conjunctive queries (UCQ) with inequalities. A UCQ with inequalities is a formula of the form $\varphi(\bar{x}) = \exists\bar{y}_1\varphi_1(\bar{y}_1, \bar{x}) \vee \cdots \vee \exists\bar{y}_n\varphi_n(\bar{y}_n, \bar{x})$, where for every $i \in \{1, \dots, n\}$:

- φ_i is a conjunction of equalities, inequalities, and positive occurrences of predicates, and
- the set of free variables in φ_i is precisely the set of free variables in φ.

Denote by UCQ$^{\neq}$ the set of all UCQs with inequalities. Before proceeding with our translation, we need to prove that positive existential formulas can be translated to UCQs with inequalities under certain conditions.

LEMMA 4. *Let $I \subseteq \mathbf{I}$ and let φ be a positive existential formula in $\mathcal{L}^I_{RDF}$. There is an $\mathcal{L}^I_{RDF}$-formula γ in UCQ$^{\neq}$ such that*

- *The predicate Dom does not occur in γ,*
- *Every equality and inequality in γ contains at least one variable,*
- *$\varphi \equiv_{RDF} \gamma$.*

PROOF. Let φ be a positive existential formula in $\mathcal{L}^I_{\text{RDF}}$. Define

$$Adom(x) = \exists y \exists z (T(x, y, z) \vee T(y, x, z) \vee T(y, z, x))$$

and let φ_T be the result of replacing in φ every occurrence of *Dom* by *Adom*. It is clear that φ_T is also a positive existential formula and that it does not mention the predicate *Dom*. It is also clear that $\varphi_T \equiv_{\text{RDF}} \varphi$, since in every structure representing an RDF graph *Dom* and *Adom* are equivalent. Since φ_T is positive existential, we can assume w.l.o.g that $\varphi_T(\bar{x}) = \exists\bar{y}_1\varphi_1(\bar{y}_1, \bar{x}_1) \vee \cdots \vee \exists\bar{y}_n\varphi_n(\bar{y}_n, \bar{x}_n)$ where each φ_i is a conjunction of equalities, inequalities, and positive occurrences of predicates (Abiteboul et al., 1995) . Notice, however, that the free variables in the disjuncts are not necessarily $\bar{x}$. Let ψ be the result of applying the next procedure over φ_T:

(1) remove every equality between two equal constants,
(2) remove every disjunct with an occurrence of T mentioning the constant n,
(3) remove every disjunct with an equality between two distinct constants.

Since equalities between equal constants are tautologies, the first operation does not alter the formula. In a structure corresponding to an RDF graph, the element associated with the constant n does not appear in any occurrence of the predicate T, so the second operation preserves the equivalence $\equiv_{\text{RDF}}$. Moreover, in structures corresponding to RDF graphs every two constants have different interpretation, and hence the third operation also preserves the equivalence in these structures. It follows that $\varphi_T \equiv_{\text{RDF}} \psi$. Finally, we transform ψ into a formula γ such that every disjunct of γ has the same free variables as ψ. Assume that $\psi(\bar{x}) = \exists \bar{y}_1 \psi_1(\bar{y}_1, \bar{x}_1) \vee \cdots \vee \exists \bar{y}_n \psi_n(\bar{y}_n, \bar{x}_n)$, where $\bar{x}$ is the set of free variables in ψ and $\bar{x}_i$ is the set of free variables in ψ_i. For $i \in \{1, \ldots, n\}$ define

$$\gamma_i(\bar{x}) = \bigvee_{X \subseteq \bar{x} \setminus \bar{x}_i} \exists \bar{y}_i \left(\psi_i(\bar{x}_i, \bar{y}_i) \wedge \bigwedge_{x \in X} Adom(x) \wedge \bigwedge_{x \in \bar{x} \setminus (\bar{x}_i \cup X)} x = n \right)$$

Finally, let $\gamma(\bar{x}) = \gamma_1(\bar{x}) \vee \cdots \vee \gamma_n(\bar{x})$. We show that $\gamma(\bar{x}) \equiv_{\text{RDF}} \psi(\bar{x})$.

- Let G be an RDF graph and let $\bar{a}$ be a tuple such that $G_{\text{FO}} \models \gamma(\bar{a})$. Hence, $G_{\text{FO}} \models \gamma_i(\bar{a})$ for some $i \in \{1, \ldots, n\}$. This implies that $G_{\text{FO}} \models \exists \bar{y}_i(\psi(\bar{a}_i, \bar{y}_i))$, where $\bar{a}_i$ is the tuple that results from restricting $\bar{a}$ to $\bar{x}_i$. By the definition of ψ, this implies that $G_{\text{FO}} \models \psi(\bar{a})$.
- Let G be an RDF graph and let $\bar{a}$ be a tuple such that $G_{\text{FO}} \models \psi(\bar{a})$. Hence, $G_{\text{FO}} \models \exists \bar{y}_i(\psi(\bar{a}_i, \bar{y}_i))$ for some $i \in \{1, \ldots, n\}$, where a_i is the tuple that results from restricting $\bar{a}$ to $\bar{x}_i$. Let X be the set of variables in $\bar{a} \setminus \bar{a}_i$ that are assigned to elements mentioned in $T^{G_{\text{FO}}}$. Since the only element not mentioned in $T^{G_{\text{FO}}}$ is N,

and N is assigned to the constant n, it is clear that

$$G_{\text{FO}} \models \exists \bar{y}_i \left(\psi(\bar{x}_i, \bar{y}_i) \wedge \bigwedge_{x \in X} Adom(x) \wedge \bigwedge_{x \in \bar{x} \setminus (\bar{x}_i \cup X)} x = n \right)$$

and hence $G_{\text{FO}} \models \gamma(\bar{a})$.

We have that γ satisfies the desired conditions. In particular, the set of free variables in every disjunct of γ is $\bar{x}$. This concludes the proof since $\varphi \equiv_{\text{RDF}} \varphi_T \equiv_{\text{RDF}} \psi \equiv_{\text{RDF}} \gamma$.

□

Having this equivalence, we can now present the main transformation from positive existential formulas to SPARQL. As previously mentioned, we are particularly interested in the syntactic form of the resulting graph pattern.

THEOREM 8. *Let φ be a positive existential formula. There is a graph pattern P in* AUF^{π} *such that $\varphi \equiv_{RDF} P$.*

PROOF. By Lemma 4 we can assume w.l.o.g. that (1) φ is in UCQ$^{\neq}$, (2) in φ the predicate *Dom* is not mentioned, (3) the constant n is not present in any occurrence of T, and (4) every equality and inequality mentions at least one variable. We proceed by transforming the disjuncts of φ into SPARQL graph patterns. Suppose that φ is the following formula

$$\varphi(\bar{x}) = \exists \bar{y}_1 \varphi_1(\bar{y}_1, \bar{x}) \vee \cdots \vee \exists \bar{y}_j \varphi_j(\bar{y}_j, \bar{x}).$$

Fix $k \in \{1, \ldots, j\}$ and assume

$$\varphi_k = T(u_1, v_1, w_1) \wedge \cdots \wedge T(u_n, v_n, w_n) \wedge$$
$$a_1 = b_1 \wedge \cdots \wedge a_m = b_m \wedge c_1 \neq d_1 \wedge \cdots \wedge c_\ell \neq d_\ell$$

Where u_i, v_i and w_i are either variables or IRIs, and a_i, b_i, c_i and d_i are either variables, IRIs or the constant n (for i in the suitable intervals). For every equality $a_i = b_i$, define the filter condition R_i piecewise as $\neg\text{bound}(?X)$ if $\{a_i, b_i\} = \{n, ?X\}$ and $a_i = b_i$ otherwise. For every inequality $c_i \neq d_i$, define the filter condition S_i piecewise as bound(?X) if $\{c_i, d_i\} = \{n, ?X\}$ and $c_i \neq d_i$ otherwise. Define the graph pattern Q_k as

$$Q_k = ((u_1, v_1, w_1) \text{ AND } \cdots \text{ AND } (u_n, v_n, w_n)) \text{ FILTER } (R_1 \wedge \cdots \wedge R_m \wedge S_1 \wedge \cdots \wedge S_\ell)$$

By the conditions of Lemma 4 this graph pattern is well-defined, and the free variables in φ_k and Q_k are exactly $\bar{x}$. We now need to prove that $\varphi_k \equiv_{\text{RDF}} Q_k$.

- [$\Rightarrow$] Let G be an RDF graph and $\mu \in [\![Q_k]\!]_G$. This implies that $\mu((u_i, v_i, w_i)) \in G$. By the definition of G_{FO} and $\mu_{\text{FO}}^{\text{var}(Q_k)}$, we have $G_{\text{FO}} \models T(u_i, v_i, w_i)$ for each $i \in \{1, \ldots, n\}$ when replacing variables according to $\mu_{\text{FO}}^{\text{var}(Q_k)}$. Now let $i \in \{1, \ldots, n\}$. Recall from Definition 5.1.4 that variables that are not binded in μ are asigned to N in $\mu_{\text{FO}}^{\text{var}(Q_k)}$. Hence, as $\mu \models R_i$ and $\mu \models S_i$, it is easy to see that $G_{\text{FO}} \models (a_i = b_i)$ and $G_{\text{FO}} \models (c_i \neq d_i)$ when replacing variables according to $\mu_{\text{FO}}^{Q_k}$. We conclude that G_{FO} satisfies each conjunct of φ_k when variables are replaced according to μ, and therefore $G_{\text{FO}} \models \varphi_k(\mu_{\text{FO}}^{Q_k})$.
- [$\Leftarrow$] Let G be an RDF graph and let μ be a mapping such that $G_{\text{FO}} \models \varphi_k(\mu_{\text{FO}}^{\text{var}(Q_k)})$. We have that $G_{\text{FO}} \models T(u_i, v_i, w_i)$ when replacing variables according to $\mu_{\text{FO}}^{\text{var}(Q_k)}$, and hence $\mu((u_i, v_i, w_i)) \in G$ for each $i \in \{1, \ldots, n\}$. Again, the variables assigned to N by $\mu_{\text{FO}}^{\text{var}(Q_k)}$ are precisely those variables not binded by μ. Hence, as $G_{\text{FO}} \models (a_i = b_i)$ and $G_{\text{FO}} \models (c_i \neq d_i)$ when replacing variables according to $\mu_{\text{FO}}^{Q_k}$, it is easy to se that $\mu \models R_i$ and $\mu \models S_i$. By the semantics of AND and FILTER , we conclude that $\mu \in [\![Q_k]\!]_G$.

We have transformed the conjunctive part of each disjunct of φ into a first-order formula. Now we need to include in our transformation the existential quantification. This is achieved by means of SELECT.

Define the pattern P_k as SELECT $\bar{x}$ WHERE Q_k. We show that $P_k \equiv_{\text{RDF}} \exists \bar{y}_k \varphi_k(\bar{y}_k, \bar{x})$. Let G be an RDF graph.

- [$\Rightarrow$] Let $\mu \in [\![P_k]\!]_G$. By the semantics of SELECT, there must be a mapping $\mu' \in [\![Q_k]\!]_G$ such that $\mu'_{|\bar{x}} = \mu$. Since $Q_k \equiv_{\text{RDF}} \varphi_k$, this implies that $G_{\text{FO}} \models \varphi_k(\mu'^{\bar{x}\cup\bar{y}_k}_{\text{FO}})$. Since the projection of $\mu'^{\bar{x}\cup\bar{y}_k}_{\text{FO}}$ to $\bar{x}$ is precisely $\mu^{\bar{x}}_{\text{FO}}$, we have that $G_{\text{FO}} \models \exists\bar{y}_k\varphi_k(\bar{y}_k, \mu^{\bar{x}}_{\text{FO}})$, concluding the first direction.
- [$\Leftarrow$] Let μ be a mapping such that $G_{\text{FO}} \models \exists\bar{y}_k\varphi_k(\bar{y}_k, \mu^{\bar{x}}_{\text{FO}})$. Then, there is a tuple a that extends $\mu^{\bar{x}}_{\text{FO}}$ by assigning an IRI or the value N to each variable in $\bar{y}_k$. Hence, a corresponds to $\mu'^{\bar{x}\cup\bar{y}_k}_{\text{FO}}$ for some mapping μ'. Since $G_{\text{FO}} \models \varphi_k(\mu'^{\bar{x}\cup\bar{y}_k}_{\text{FO}})$ and $\varphi_k \equiv_{\text{RDF}} Q_k$, this means that $\mu' \in [\![P_k]\!]_G$. As the restriction of μ' to $\bar{x}$ is μ, we have that $\mu \in [\![\text{SELECT}\,\bar{x}\ \text{WHERE}\ P_k]\!]_G$.

Having for each disjunct φ an equivalent graph pattern, we finally proceed to create a graph pattern that is equivalent to φ. This graph pattern is defined, as expected, as the dijunction between the previously constructed patterns. Let $P = P_1$ UNION $\cdots$ UNION P_j. It is immediate to prove that $P \equiv_{\text{RDF}} \varphi$: Let G be an RDF graph. A mapping μ belongs to $[\![P]\!]_G$ if an only if there is a $k \in \{1, \ldots, j\}$ such that $\mu \in [\![P_k]\!]_G$. We already proved this is the case if and only if there is a $k \in \{1, \ldots, j\}$ such that $G_{\text{FO}} \models \exists\bar{y}_k\varphi_k(\bar{y}_k, \mu^{\bar{x}}_{\text{FO}})$, concluding the proof.

$\square$

At this point we have one transformation from SPARQL to FO and one transformation from FO to graph patterns in AUF^{π}. Unfortunately, the composition of these two transformations does not preserve all the answers, but only the maximal ones. Notice that this is expected, as it is well known that graph patterns in AUF^{π} are not only weakly-monotone, but also monotone. However, we will see in the next chapters that maximal answers play an important role in the evaluation of graph patterns. We say that two graph patterns P and Q are equivalent in maximal answers if for every graph G, the set of maximal mappings (w.r.t. $\preceq$) in $[\![P]\!]_G$ coincides with the set of maximal mappings in $[\![Q]\!]_G$.

THEOREM 9. *Let P be a weakly-monotone SPARQL graph pattern. There is an graph pattern Q in* AUF^{π} *that is equivalent in maximal answers to P.*

PROOF. Let P be a weakly-monotone SPARQL graph pattern. Let φ be the existential first-order formula such that $P \equiv_{\text{MAX}} \varphi$ obtained from applying Theorem 3. Denote by ψ the UCQ with inequalities equivalent to φ constructed by applying Theorem 4. Now let Q be the graph pattern such that $Q \equiv_{\text{RDF}} \psi$ obtained from applying Theorem 8. We prove that Q is equivalent in maximal answers to P.

- [$\Rightarrow$] Let G be an RDF graph and let μ be a maximal mapping in $[\![P]\!]_G$. Then, we know that $G_{\text{FO}} \models \varphi(\mu^P_{\text{FO}})$ and hence $G_{\text{FO}} \models \psi(\mu^P_{\text{FO}})$. It readily follows that $\mu \in [\![Q]\!]_G$.
- [$\Leftarrow$] Let G be an RDF graph and let μ be a maximal mapping in $[\![Q]\!]_G$. This implies that μ^P_{FO} is a maximal tuple (w.r.t. $\preceq$) such that $G_{\text{FO}} \models \psi(\mu^P_{\text{FO}})$. Since ψ and P are equivalent in maximal answers, we conclude that $\mu \in [\![P]\!]_G$.

$\square$

The previous theorem concludes our application of interpolation theorems to weak-monotonicity over SPARQL. It is interesting to mention that this theorem can be easily proved for well-designed graph patterns, but it is not clear how to prove it over weakly-monotone graph patterns without going through the presented translation to first-order logic. In the next two chapters we will see that this theorem is a powerful tool for characterizing and capturing semantic properties over different fragments of SPARQL. It is important to mention that the theorem provides, for every weakly-monotone pattern P, a new graph pattern Q that is monotone. Hence, we will need some further work we to deduce properties over weakly-monotone graph patterns.

6. SYNTACTIC CHARACTERIZATIONS IN SPARQL

After studying how interpolation is used to establish preservation theorems, we have managed to build a relation between SPARQL and FO. From this relation we proved that every weakly-monotone graph pattern is equivalent in maximal answers to an OPT-free graph pattern. In this chapter we study the main consequences of this result, focusing on finding weakly-monotone fragments that are more expressive than well-designed graph patterns, as motivated in Chapter 3. We study the expressive power of different fragments of SPARQL, and introduce an operator that replaces OPT with a different way of obtaining optional information.

6.1. Removing subsumption: the NS operator

Theorem 9 asserts that given a graph pattern P, there is a graph pattern Q in AUF^{π} that is equivalent to P in maximal answers. However, Q provides no information about what subsumed mappings could P output. Therefore, it is natural to start our study by focusing on graph patterns that never output subsumed mappings. We say that a graph pattern P is subsumption-free whenever for every RDF graph G and every two mappings $\mu_1, \mu_2 \in [\![P]\!]_G$, it is the case that $\mu_1 \not\prec \mu_2$.

Assume a graph pattern P is subsumption-free, and Q is the result from applying Theorem 9 to P. Hence, P and Q are equivalent in maximal answers. But since P is subsumption-free, the maximal mappings coming from Q are exactly the set of mappings coming from P. Thus, if we defined an operation for removing subsumed mappings, we could retrieve all the information of P from Q. It is interesting to notice that this operation naturally arises from the applications of interpolation theorems, but at the same time is just a different way of obtaining optional information. In fact, an operator for obtaining maximal answers called *minimal union relation* was already studied in the context of incomplete information under relational databases (Galindo-Legaria, 1994). We introduce a new operator for querying RDF datasets by extending the syntax and semantics of SPARQL:

DEFINITION 6.1.1. *If P is a SPARQL graph pattern, then* $\mathrm{NS}(P)$ *is a graph pattern. Moreover, given an RDF graph G,*

$$[\![\mathrm{NS}(P)]\!]_G = \{\mu \in [\![P]\!]_G \mid \textit{there is no } \mu' \in [\![P]\!]_G \textit{ such that } \mu \prec \mu'\}.$$

For a given fragment of SPARQL (e.g, AUF^{π}), denote the extension of such fragment with the NS operator by adding 'NS' as a subscript (e.g, $\mathsf{AUF}^{\pi}_{\mathrm{NS}}$). It can be seen from the previous discussion that whenever we have a subsumption-free graph pattern, we can obtain an equivalent OPT-free graph pattern that only uses the NS operator at the top-most level. The following fragment is therefore a natural set for capturing the set of subsumption-free weakly-monotone graph patterns:

DEFINITION 6.1.2. *A graph pattern is called simple if it is of the form $NS(P)$, where P is in* AUF^{π}.

As already mentioned, we are interested in finding fragments that only contain weakly-monotone graph patterns. We show that this is the case for simple patterns.

LEMMA 5. *Every simple graph pattern is weakly-monotone.*

PROOF. It is trivial to show that if Q is a weakly-monotone graph pattern, then $\mathrm{NS}(Q)$ is weakly-monotone. Since every AUF^{π} graph pattern is weakly-monotone, it immediately follows that every simple pattern is also weakly-monotone. □

We now proceed to study the expressive power of simple patterns.

6.2. Simple patterns and subsumption-free patterns

A first consequence of the definition of simple patterns and Theorem 9 is that this fragment covers the fragment of all subsumption-free weakly-monotone graph patterns.

THEOREM 10. *Let P be a subsumption-free weakly-monotone SPARQL graph pattern. Then, there is a simple pattern that is equivalent to P.*

PROOF. Let P be a subsumption-free weakly-monotone SPARQL graph pattern, and let Q be the AUF^{π} graph pattern that result of applying Theorem 9 to P. We know that P and Q are equivalent in maximal answers. But since P only outputs maximal mappings, we have that for every RDF graph G and every mapping μ, it is the case that $\mu \in [\![P]\!]_G$ if and only if μ is a maximal mapping (w.r.t. $\preceq$) in $[\![Q]\!]_G$. As the maximal mappings in $[\![Q]\!]_G$ are precisely the mappings in $[\![\text{NS}(Q)]\!]_G$, we obtain that P is equivalent to $\text{NS}(Q)$, which is a simple pattern. □

Recall that in Chapter 3, we proved that the fragment of well-designed graph patterns (including disjunction at the top-most level) does not cover weak-monotonicity over SPARQL. It is interesting to notice that the counterexample used in that prove is actually subsumption-free:

$$(?X, a, b)\ \text{OPT}\ ((?X, c, ?Y)\ \text{UNION}\ (?X, d, ?Z)).$$

This graph pattern is equivalent to

$$\begin{aligned}\text{NS}\,[(?X, a, b)\ \text{UNION}\ ((?X, a, b)\ \text{AND}\ (?X, c, ?Y))&\\ \text{UNION}\ ((?X, a, b)\ \text{AND}\ (?X, d, ?Z))],&\quad (6.1)\end{aligned}$$

which is a simple pattern. We obtain as a corollary that there are simple patterns that cannot be expressed as well-designed graph patterns.

On the other hand, from Theorem 10 we can directly deduce that weakly-monotone graph patterns in AOF can be translated into simple patterns (given that AOF is subsumption-free (Pérez et al., 2009)).

COROLLARY 2. *The fragment of UNION-free well-designed graph patterns is properly contained in the fragment of simple patterns.*

We would also like to establish containment in the opposite direction, showing that fragments mentioning NS can be translated into fragments of SPARQL. To this end, we start by showing a more general result: SPARQL and $\mathsf{AUF}^{\pi}_{\mathrm{NS}}$ have the same expressive power. It is trivial to prove that SPARQL is contained in $\mathsf{AUF}^{\pi}_{\mathrm{NS}}$, given that P_1 OPT P_2 is equivalent to $NS(P_1 \text{ UNION } (P_1 \text{ AND } P_2))$. To prove the other direction, we make use of the fact that every graph pattern can be translated into an equivalent pattern in UNION-normal-form. A graph pattern P is said to be in UNION-normal-form if P is a disjunction of UNION-free graph patterns. In (Pérez et al., 2006a) the authors proved that every graph pattern in AUFO can be transformed into UNION-normal-form. We extend this proof to show that this is also the case for SPARQL graph patterns including SELECT.

LEMMA 6. *Every SPARQL graph pattern is equivalent to a graph pattern in UNION-normal-form.*

PROOF. Let P be a SPARQL graph pattern. We proceed by induction over the structure of P. We only consider the case in which $P = \text{SELECT}\, V \text{ WHERE } Q$, as the other cases have already been completed in the original proof[1] for AUOF (Pérez et al., 2006a).

Assume $P = \text{SELECT}\, V \text{ WHERE } Q$. By induction hypothesis we can assume that $Q = Q_1 \text{ UNION } \cdots Q_n$ where each disjunct is UNION-free. We prove that

$$P \equiv (\text{SELECT}\, V \text{ WHERE } Q_1) \text{ UNION } \cdots \text{ UNION } (\text{SELECT}\, V \text{ WHERE } Q_n).$$

Let G be an RDF graph and μ be a mapping.

[1]The original proof had an issue in the OPT case, it was later corrected in errata by the authors.

- [$\Rightarrow$] Assume $\mu \in [\![P]\!]_G$. Then, there is a mapping $\mu' \in [\![Q]\!]_G$ such that $\mu'_{|V} = \mu$. implies there is an $i \in \{1, \ldots, n\}$ such that $\mu' \in [\![Q_i]\!]_G$. It follows that $\mu \in [\![\text{SELECT}\, V \ \text{WHERE} \ Q_i]\!]_G$.
- [$\Leftarrow$] Assume $\mu \in [\![\text{SELECT}\, V \ \text{WHERE} \ Q_i]\!]_G$ for $i \in \{1, \ldots, n\}$. Then, there is a mapping $\mu' \in [\![Q_i]\!]_G$ such that $\mu'_{|V} = \mu$. It follows that $\mu' \in [\![Q]\!]_G$, and hence $\mu \in [\![P]\!]_G$.

□

To prove that $\mathsf{AUF}^{\pi}_{\text{NS}}$ is contained in SPARQL we actually make use of a stronger version of UNION-normal-form. We abuse notation by writing $D \in P$ whenever P is a graph pattern in UNION-normal-form and D is a disjunct of P.

LEMMA 7. *Let P be a SPARQL graph pattern. Then, there is a graph pattern P' in UNION-normal-form that is equivalent to P, and that satisfies the following condition: for every $D \in P'$ there is a set of variables $V_D \subseteq var(P)$ such that, for every RDF graph G and every $\mu \in [\![D]\!]_G$, it is the case that $dom(\mu) = V_D$.*

PROOF. Let P be a SPARQL graph pattern. For every $V \subseteq \text{var}(P)$, define the graph pattern

$$P_V = P\,\text{FILTER}\left(\bigwedge_{?X \in V} \text{bound}(?X) \wedge \bigwedge_{?X \in \text{var}(P) \setminus V} \neg\text{bound}(?X)\right).$$

Notice that for every graph pattern G, $[\![P_V]\!]_G$ is the set of mappings μ in $[\![P]\!]_G$ such that $\text{dom}(\mu) = V$. Now, define the pattern P'_V as the transformation of P_V into UNION-normal-form. It is clear that the domain of every mapping that comes from the disjuncts of P'_V must be exactly V. Define P' as the disjunction (by means of UNION) of every P'_V ($V \subseteq \text{var}(P)$). We prove that P is equivalent to P'. Let G be an RDF graph and $\mu \in [\![P]\!]_G$. It is clear that $\mu \in [\![P_{\text{dom}(\mu)}]\!]_G$, and hence $\mu \in [\![P'_{\text{dom}(\mu)}]\!]_G$, which implies $\mu \in [\![P']\!]_G$. For the converse, let $\mu \in [\![P']\!]_G$. There is a set V of variables such that $\mu \in [\![P'_V]\!]_G$, and hence $\mu \in [\![P_V]\!]_G$, which implies $\mu \in [\![P]\!]_G$. Finally, as P' is a disjunction of graph patterns in UNION-normal-form, P' is also in UNION-normal-form. Moreover, the disjuncts in the UNION-normal-form version of P_V can only output mappings whose domain is precisely V, which concludes the proof. □

In simple words the previous lemma allows us to distinguish what are the variables that bind the mappings coming from each disjunct. In the following we use the operation MINUS , which is defined as follows:

$$P_1 \ \mathsf{MINUS} \ P_2 = (P_1 \ \text{OPT} \ (P_2 \ \text{AND} \ (?x_1, ?x_2, ?x_3))) \, \text{FILTER} \ \neg bound(?x_1).$$

Given an RDF graph G, P_1 MINUS P_2 retrieves the mappings in $[\![P_1]\!]_G$ that are not compatible with any mapping in $[\![P_2]\!]_G$.

Now we are ready to prove the containment in the remaining direction.

LEMMA 8. *Every* AUF^{π}_{NS} *graph pattern can be transformed into an equivalent SPARQL pattern.*

PROOF. Let P be a graph pattern in $\mathsf{AUF}^{\pi}_{\text{NS}}$. We proceed by induction over the structure of P. The basic case is trivial as a triple pattern is already in SPARQL. For the inductive step to succeed, we actually need to prove a stronger property: every graph pattern in $\text{SPARQL}_{\text{NS}}$ can be translated into SPARQL. Assume $P = NS(Q)$, which is the only nontrivial case. By hypothesis we can assume Q is in SPARQL (it might mention OPT). From Lemma 7 we can suppose that Q is in UNION-normal-form and, moreover, that each disjunct of Q can only output mappings binding a fixed set of variables V_Q. Let Q' be a disjunct of Q, and assume $Q_1, \ldots, Q_n$ are all disjuncts of Q such that $V_{Q'} \subsetneq V_{Q_i}$. Define the graph pattern

$$Q'_{\text{NS}} = Q' \ \mathsf{MINUS} \ (Q_1 \ \text{UNION} \ \cdots \ \text{UNION} \ Q_n).$$

We prove that $NS(Q)$ is equivalent to the graph pattern defined as

$$R = \underset{Q' \in Q}{\text{UNION}} \ Q'_{\text{NS}}.$$

- [$\Rightarrow$] Let G be an RDF graph and let $\mu \in [\![NS(Q)]\!]_G$. We have that $\mu \in [\![Q']\!]_G$ for some disjunct Q' of Q, and that there is no mapping $\mu' \in [\![NS(Q)]\!]_G$ subsuming μ. It follows that there is no disjunct Q'' of Q and mapping $\mu'' \in [\![Q'']\!]_G$ such that $V_{Q'} \subsetneq V_{Q''}$ and $\mu \sim \mu''$, from which we can deduce that $\mu \in [\![Q'_{\text{NS}}]\!]_G$.
- [$\Leftarrow$] Let G be an RDF graph and let $\mu \in [\![R]\!]_G$. We have that $\mu \in [\![Q'_{\text{NS}}]\!]_G$ for some disjunct Q' of Q. Then, $\mu \in [\![Q']\!]_G$ and there is no disjunct Q'' of Q and mapping $\mu'' \in [\![Q'']\!]_G$ such that $V_{Q'} \subsetneq V_{Q''}$ and $\mu \sim \mu'$. This obviously implies that there is no mapping in $[\![Q]\!]_G$ subsuming μ, and hence $\mu \in [\![NS(Q)]\!]_G$.

This concludes the proof as the inductive step is trivial in the other cases. □

From the previous proof we conclude that the NS operator can be removed by introducing MINUS, which is defined in terms of OPT. With this result we can finally deduce the aforementioned equivalence as a corollary:

COROLLARY 3. *The languages SPARQL and* AUF^{π}_{NS} *are equivalent in expressive power.*

This equivalence has some further implications. Given a simple pattern P, we can now construct a SPARQL graph pattern Q that is equivalent to P. But since P is subsumption-free, Q is also subsumption-free. Hence, we can deduce the following

COROLLARY 4. *The fragment of simple patterns is equivalent in expressive power to the fragment of subsumption-free weakly-monotone graph patterns.*

6.3. Weak-monotonicity and ns-patterns

Now that we have characterized the fragment of weakly-monotone graph patterns in which subsumption is not allowed, we proceed to allow subsumption by means of disjunction at the top-most level.

DEFINITION 6.3.1. *A graph pattern P in* AUF^{π}_{NS} *is an* ns-pattern *if it is a disjunction of simple patterns.*

It is trivial to see that all ns-patterns are weakly-monotone, as they are a disjunction of weakly-monotone graph patterns. It is also easy to see that the fragment of ns-patterns is actually more expressive than simple patterns, as simple patterns are subsumption-free. As a final result of this chapter, from the definition of ns-patterns and Corollary 4 we obtain the following.

COROLLARY 5. *The fragment of ns-patterns has the same expressive power as the fragment of disjunctions of subsumption-free weakly-monotone graph patterns.*

So far, we have shown that the fragment of well-designed patterns falls short in terms of expressive power, and we presented a new operator that allows us to define more expressive syntactic fragments of weakly-monotone graph patterns. Moreover, we found the semantic notions characterized by those fragments. These results present substantial advances in out understanding of the expressive power of weakly-monotone SPARQL graph patterns. However, they are not as general as we would like, since the captured fragments include in their definitions the notion of being subsumption-free. Weak-monotonicity is a desired feature in SPARQL (as we have widely discussed) but disallowing subsumption is a restriction that arises as a necessity for finding characterizations from applying interpolation. This is actually what prevents us from finding a syntactic fragment that captures the set of all weakly-monotone graph patterns, problem that remains open. Actually, the problem of whether ns-patterns characterize weak-monotonicity is also open.

Provided the tools we have developed, one could try to directly find a fragment characterizing weak-monotonicity by using the following method: Given a weakly-monotone graph pattern P, construct a weakly-monotone graph pattern Q such that for every RDF graph G, the mappings in $[\![P]\!]_G$ corresponds to the maximal mappings in $[\![Q]\!]_G$. This correspondence could be defined, for example, by adding new variables to Q in a way such that every mapping in $[\![P]\!]_G$ can be obtained by projecting a maximal mapping in $[\![Q]\!]_G$ to $\text{var}(P)$. This would imply the equivalence $P \equiv \text{SELECT}\,\text{var}(P)\ \text{WHERE}\ (\text{NS}(Q))$. Then, by applying Theorem 9 we could construct a graph pattern Q' in AUF^{π} that is equivalent in maximal answers to P, which would imply $P \equiv \text{SELECT}\,\text{var}(P)\ \text{WHERE}\ (\text{NS}(Q'))$. Therefore, this syntactic form would be a characterization of weak-monotonicity. This procedure was performed several times using different types of correspondences, but the touchstone was to define a procedure that given a graph pattern P constructs a weakly-monotone graph pattern Q, such that the maximal mappings from Q correspond to all the results from P.

In the next chapter we will argue that the notion of being subsumption-free is only necessary because graph patterns output table-like structures (mappings) while querying graph-like structures. While this condition is inherent to SPARQL graph patterns, there is also a class of queries that output RDF graphs. We will formalize these queries, and study the corresponding semantic properties they should satisfy to conform to the OWA. Under this new class of queries, we are actually able to find much more general results.

7. A LANGUAGE PRODUCING RDF GRAPHS

In the previous chapter, we characterized semantic properties of interest using the NS operator and simple syntactic fragments. Unfortunately, we were unable to find a query language that captures weak-monotonicity, mainly because when applying interpolation it is not clear how to maintain the subsumed answers. In this chapter, we formalize a different set of SPARQL queries. These queries take RDF graphs as input and produce, instead of mappings, RDF graphs as output. This is the set of CONSTRUCT queries. We show that these queries have several novel properties, and actually we are able to characterize the fragment that conforms to the open-world assumption. According to this result, we present the syntactic fragment of this characterization as a proper language for querying RDF graphs.

7.1. RDF graphs as input, mappings as output

In this section we discuss what are the consequences of querying RDF graphs and generating sets of mappings as output. This will later serve as a motivation for defining the fragment of CONSTRUCT queries. Assume P is a SPARQL$_{\text{NS}}$ graph pattern and G is an RDF graph. We start by discussing two consequences from the definition of $[\![P]\!]_G$. The first is that the set of mappings $[\![P]\!]_G$ is not a concise representation of the information obtained from querying G with P, and the second is that no relation can be established between the resources mentioned in $[\![P]\!]_G$ by only inspecting $[\![P]\!]_G$. To illustrate these issues, we present a simple example.

Example 5. *Let G be the RDF graph presented in Figure 7.1. Assume that we want to query for the names of professors and the establishments they work at, and optionally (only if available) their emails. To get this information, we would write the following*

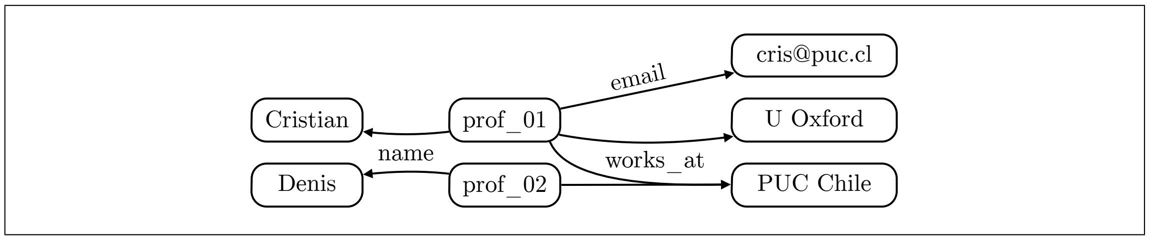

FIGURE 7.1. An RDF graph containing information about academic staff.

SPARQL graph pattern:

$$P = \text{SELECT } \{?n, ?e, ?u\} \text{ WHERE}$$
$$(((?p, name, ?n) \text{ AND } (?p, works\_at, ?u)) \text{ OPT } (?p, email, ?e))$$

The answer $[\![P]\!]_G$ *consists of three mappings, illustrated in Table 7.1.*

TABLE 7.1. Tabular representation of mappings in the answer to a graph pattern.

	?n	*?u*	*?e*
μ_1	*Denis*	*PUC Chile*	
μ_2	*Cristian*	*U Oxford*	*cris@puc.cl*
μ_3	*Cristian*	*PUC Chile*	*cris@puc.cl*

This example is a simple use case of SPARQL, but it correctly illustrates the two problems mentioned above. We first explain why the set of mappings $[\![P]\!]_G$ in the example is not a concise representation of the obtained information. Consider for example the URI *cris@puc.cl*. This URI is mentioned twice in a single column, yet both occurrences represent the same fact, namely that *cris@puc.cl* is the email of *Cristian*. Hence, mentioning this URI twice does not provide more information than mentioning it once. Notice, moreover, that all the information in $[\![P]\!]_G$ can be obtained from the RDF graph in Figure 7.2. Thus, if we were able to produce graphs as answers, in this case we would be able to obtain all the information from $[\![P]\!]_G$ in a more concise representation.

Now we discuss the second problem, namely that no relation can be established between the resources in the answer to P over G by only looking at the set of mappings in

$[\![P]\!]_G$. Suppose we have no information about G or P, and we are presented set of mappings $[\![P]\!]_G$. What can we deduce? What is the relation between the URIs? Does *Denis* work at *PUC Chile*? Does he have a friend working there? None of these questions can be answered by only looking at $[\![P]\!]_G$. Furthermore, it is clear that this cannot be solved by modifying P, as the same questions about the URIs would remain unanswered. In order to make sense of the mappings, it is necessary to look at the graph pattern and/or at the original data. However, if we were able to output RDF graphs instead of mappings, by only looking at the output we would be able to understand what is the relation between the resources, as depicted in Figure 7.2.

There are also further practical consequences of producing mappings when querying RDF graphs: while recursive queries have been part of SQL for more than twenty years, we are still left without a comprehensive operator to define recursive queries in SPARQL. The version 1.1 of SPARQL includes the property paths primitive (W3C SPARQL Working Group, 2013), but this additional feature is very restrictive in expressing recursive queries (Libkin, Reutter, & Vrgoč, 2013). It is then important to define a query language that outputs RDF graphs, so recursion can be applied over RDF graphs.

Being conscious of the issues presented above, in the next section we introduce the fragment of SPARQL queries generating RDF graphs as output.

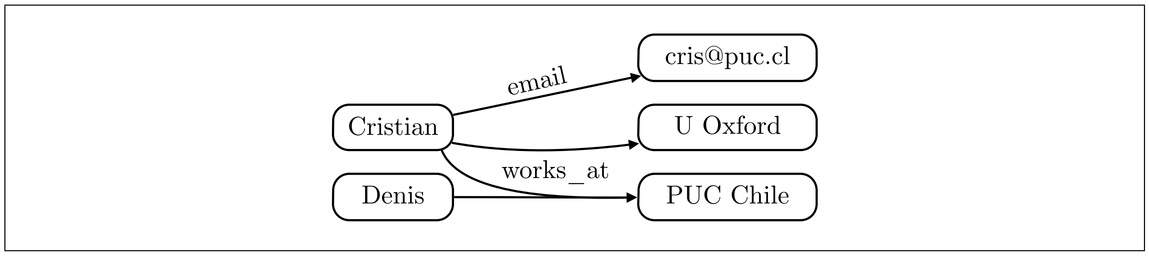

FIGURE 7.2. An RDF graph representing mappings from Table 7.1. This representation is more concise, and the structure itself relates the mentioned resources.

7.2. Formalizing CONSTRUCT queries

CONSTRUCT queries in SPARQL shape the class of effective queries whose inputs and answers are RDF graphs, so it is conceivable that these queries do not present the problems mentioned in the previous section. As mentioned throughout this dissertation, a good deal of research has been devoted to the study of SPARQL graph patterns. Surprisingly, and despite being part of the official SPARQL standard, CONSTRUCT queries have received very little attention from the theoretical community. They have been proposed from a practical point of view as a language for further applications of SPARQL (Polleres, Scharffe, & Schindlauer, 2007; Bry et al., 2010), but have not been studied as a primitive in the core of SPARQL. This can be partially explained by the fact that, as some examples might suggest, the difference between CONSTRUCT queries and graph patterns seems negligible. However, as we show in the following sections, this resemblance is rather deceptive, and in many cases these queries have different properties. For example, the output of a CONSTRUCT query can mention IRIs that are not present in the queried graph, which is not possible using graph patterns. We introduce the formal definition of CONSTRUCT queries.

DEFINITION 7.2.1. *Let P be a SPARQL graph pattern and let H be a finite set of triple patterns. Then* $q =$ CONSTRUCT H WHERE P *is a CONSTRUCT query, or c-query for short. P and H are called the graph pattern and the template of q, respectively. Moreover, for ever RDF graph G the answer to q over G is defined by*

$$ans(q, G) = \{\mu(t) \mid \mu \in [\![P]\!]_G,\ t \in H, \text{ and } var(t) \subseteq dom(\mu)\ t \in H\}.$$

Let G be the RDF graph presented in Figure 7.1. We illustrate the semantics of CONSTRUCT by querying for the same information as in Example 5. We want to obtain, from the graph in Figure 7.1, the names and establishments of the academic staff, and

TABLE 7.2. Answers to the graph pattern of a c-query. The variable $?p$ occurs in all mappings but is never mentioned in the template.

	?p	?n	?u	?e
μ_1	prof_02	Denis	PUC Chile	
μ_2	prof_01	Cristian	U Oxford	cris@puc.cl
μ_3	prof_01	Cristian	PUC Chile	cris@puc.cl

optionally append their emails. We achieve this with the following c-query:

$$q = \text{CONSTRUCT } \{(?n, \mathit{works\_at}, ?u), (?n, \mathit{email}, ?e)\} \text{ WHERE}$$
$$(((?p, \text{name}, ?n) \text{ AND } (?p, \text{works\_at}, ?u)) \text{ OPT } (?p, \mathit{email}, ?e))$$

The evaluation of the graph pattern of q over G is presented in Table 7.2. Notice that the mappings are the same as in Table 7.1, but extended with a new variable $?p$. Then, to evaluate q we need to match those mappings against the triple patterns in the template of q, and produce the corresponding RDF triples. The resulting RDF graph is displayed in Figure 7.2. Notice that the triple $(\mathit{Cristian}, \mathit{email}, \mathit{cris@puc.cl})$ is generated by two different mappings matching the same triple. However, it occurs as a single triple in the output graph, which makes the answer more concise than the raw set of mappings. Notice also that we can deduce how the resources are related to each other just by looking at the output.

It is important to mention that in this dissertation we only study the fragment of c-queries without blank nodes. As opposed to graph patterns, blank nodes in c-queries produce an important difference in the semantics. Hence, the results presented here do not extend to c-queries, particularly to those in which blank nodes occur in the template.

7.3. Monotonicity and the open-world assumption

In Chapter 3, we presented the importance of having open-world semantics in query languages for the Semantic Web. We widely discussed that, when querying web data, it is fundamental to make no assumption about unavailable data. As opposed to SQL and other well-studied query languages, we concluded that monotonicity is not the correct notion for capturing the set of patterns conforming to the open-world assumption. This occurs because in the answer to a graph pattern, the atomic piece of information is an assignment, yet the answer itself is not a set of assignments but a set of mappings.

That is not the case for c-queries. The answer to a c-query is an RDF graph, and in an RDF graph the atomic piece of information is indeed an RDF triple. Hence, the same argument that established weak-monotonicity in graph patterns, namely preserving information under data extensions, now applies to monotonicity of c-queries: Assume that q is a c-query and that G_1 and G_2 are two RDF graphs such that $G_1 \subseteq G_2$. If $\text{ans}(q, G_1)$ contains a triple t that is not in $\text{ans}(q, G_2)$, when evaluating q over G_1 we were assuming that triples in $G_2 \setminus G_1$ were false, as if this was not the case then t would not have been in $\text{ans}(q, G_1)$. Thus, in c-queries we are interested in preserving triples, which is precisely captured by monotonicity.

DEFINITION 7.3.1. *A c-query q is monotone if for every two RDF graphs G_1 and G_2 such that $G_1 \subseteq G_2$, it is the case that $ans(q, G_1) \subseteq ans(q, G_2)$.*

As in the case of weak-monotonicity for graph patterns, this definition provides no insight about how to find a syntactic characterization of monotone c-queries. Moreover, the problem of deciding whether a construct query q is monotone is undecidable. However, and as opposed to the case of weak-monotonicity, the problem of characterizing monotone c-queries is actually solved by using the same interpolation techniques we used for weak-monotonicity. This is mainly a consequence of the fact that to evaluate a c-query it is only necessary to look at the maximal mappings in the evaluation of the corresponding graph pattern.

LEMMA 9. *Let $q =$ CONSTRUCT H WHERE P be a c-query. Then, q is equivalent to* CONSTRUCT H WHERE $\text{NS}(P)$.

PROOF. Let q = CONSTRUCT H WHERE P be a c-query and let G be an RDF graph. It is immediate to prove that

$$\text{ans}(\text{CONSTRUCT } H \text{ WHERE NS}(P), G) \subseteq \text{ans}(q, G),$$

since we know that $[\![NS(P)]\!]_G \subseteq [\![P]\!]_G$. For the other direction, let $\mu(t)$ be a triple in ans(q, G), where $\mu \in [\![P]\!]_G$ and $t \in H$. Since $\mu \in [\![P]\!]_G$, there must be a mapping $\mu' \in [\![NS(P)]\!]_G$ such that $\mu \preceq \mu'$. As $\mu \preceq \mu'$ and var$(t) \subseteq$ dom(μ), we know that $\mu(t) = \mu'(t)$. Therefore $\mu(t)$ is in ans(CONSTRUCT H WHERE NS$(P), G)$. □

Recall from Theorem 9 that given a SPARQL weakly-monotone graph pattern P, there is a graph pattern $Q \in \mathsf{AUF}^\pi$ that is equivalent to P in maximal answers. The previous lemma tells us that for c-queries it is sufficient to only preserve maximal answers. However, to apply Theorem 9 we need a weakly-monotone graph pattern, and there are c-queries that are monotone but their graph patterns are not weakly-monotone. The following lemma shows that this does not prevents us from our main goal.

LEMMA 10. *For every monotone c-query q, there is a template H and a weakly-monotone SPARQL graph pattern P such that*

$$q \equiv \text{CONSTRUCT } H \text{ WHERE } P.$$

PROOF. Let q = CONSTRUCT H WHERE P be a c-query. We can assume without loss of generality that var$(H) \subseteq$ var(P), as every triple in H mentioning a variable not occuring in P can be safely removed. For every triple pattern $t \in H$ define a renaming function $\sigma_t : \mathbf{V} \to \mathbf{V}$ in a way such that:

- For every $t, s \in H$ and every $v_1, v_2 \in$ var(P), it is the case that $\sigma_t(v_1) \neq \sigma_s(v_2)$.
- For every $t \in H$ and every $v \in$ var(P), $\sigma_t(v) \notin$ var(P).

For a mapping μ and a triple $t \in H$, define $\sigma_t[\mu]$ as the mapping that results from replacing the domain of μ by its image under σ_t. For every $t \in H$ let *Adom*(t) be the conjunction (by means of AND) of $Adom(?X)$ for each variable $?X$ in $\text{var}(t)$. If t has no variables then $Adom(t)$ is considered to be a tautology.

For every $t \in H$ define the pattern P^t as the result of replacing in P every occurrence of a variable $?X$ by $\sigma_t(?X)$. For every two triples $t = (t_1, t_2, t_3)$ and $s = (s_1, s_2, s_3)$ in H define $R_{t,s}$ as the filter condition $(t_1 = \sigma_s(s_1) \wedge t_2 = \sigma_s(s_2) \wedge t_3 = \sigma_s(s_3))$, assuming, for the sake of simplicity, that $\sigma_s(a) = a$ for every $a \in \mathbf{I}$. Now we define the set of graph patterns that will serve as a basis for our construction. For each $t \in H$, define P_t as

$$\text{SELECT}\,\text{var}(t)\ \text{WHERE}$$
$$\left(\left[P\ \text{UNION}\ \underset{s\in H\setminus\{t\}}{\text{UNION}}\,[(P^s\ \text{AND}\ Adom(t))\,\text{FILTER}\ R_{t,s}]\right]\text{FILTER}\ (bound(\text{var}(t)))\right)$$

We prove that the next three properties hold for every $t \in H$:

(1) For every graph G and every mapping $\mu \in [\![P]\!]_G$, if $\mu(t) \in \text{ans}(q, G)$, then $\mu(t) \in \text{ans}(\text{CONSTRUCT}\ t\ \text{WHERE}\ P_t, G)$.

(2) For every graph G, $\text{ans}(\text{CONSTRUCT}\ t\ \text{WHERE}\ P_t, G) \subseteq \text{ans}(q, G)$.

(3) P_t is weakly-monotone.

The first property immediately follows from the fact that P is one of the disjuncts of P_t, as if $\mu(t) \in \text{ans}(q, G)$, then the variables in t are bounded by μ.

Now we proceed with (2). Let G be an RDF graph and let μ be a mapping in $[\![P_t]\!]_G$ such that $\mu(t) \in \text{ans}(\text{CONSTRUCT}\ t\ \text{WHERE}\ P_t, G)$. Hence, μ must come from one of the disjuncts in P_t. If that disjunct is P, then we have that μ is the projection over $\text{var}(t)$ of a mapping in $[\![P]\!]_G$, and hence $\mu(t) \in \text{ans}(q, G)$. If not, then there is an $s \in H$ such that μ is subsumed by a mapping $\mu' \in [\![P^s\ \text{AND}\ Adom(t)\,\text{FILTER}\ R_{t,s}]\!]_G$. Then μ' is the join between two mappings. Let μ_s be the mapping in $[\![P^s]\!]_G$ of such join. Since P^s equals P by a renaming of all variables, the mapping $\sigma_s^{-1}[\mu_s]$ belongs to $[\![P]\!]_G$. Moreover, by the filter condition $R_{t,s}$, we know that $\sigma_s^{-1}[\mu_s]$ must bind all variables in $\text{var}(s)$, and hence

$\sigma_s^{-1}[\mu_s](s) \in \text{ans}(q, G)$. But from the filter condition we know that $\sigma_s^{-1}[\mu_s](s)$ equals $\mu(t)$, and hence $\mu(t)$ belongs to $\text{ans}(q, G)$, which was to be shown.

Finally we prove that P_t is weakly monotone. Let G be an RDF graph and $\mu \in [\![P_t]\!]_G$. We know that $\text{dom}(\mu) = \text{var}(t)$, and hence $\mu(t) \in \text{ans}(\text{CONSTRUCT } t \text{ WHERE } P_t, G)$. By property 2 this implies that $\mu(t) \in \text{ans}(q, G)$. Let G' be an RDF graph such that $G \subseteq G'$. Since q is monotone, there must be a triple $s \in H$ and a mapping $\mu_s \in [\![P]\!]_{G'}$ such that $\mu_s(s) = \mu(t)$. Hence $\sigma_s[\mu_s] \in [\![P^s]\!]_{G'}$. Moreover, since $\mu_s(s) = \mu(t)$, we have that $\sigma_s[\mu_s] \bowtie \mu$ satisfy $R_{t,s}$. Hence, $\sigma_s[\mu_s] \bowtie \mu$ belongs to $[\![P^s \text{ AND } \mathit{Adom}(t) \text{ FILTER } R_{t,s}]\!]_{G'}$, and hence $\mu \in [\![P_t]\!]_{G'}$. This actually tells us that P_t is monotone, and therefore weakly-monotone.

Having defined the patterns P_t and proved the three properties above, we proceed with the main result. First, define for each $t \in H$ the c-query q_t as $\text{CONSTRUCT } t' \text{ WHERE } P'_t$, where t' and P'_t are the result of renaming the variables in t and P_t, respectively, by a single function. Without loss of generality we can assume that for $t, s \in H$, the queries q_t and q_s have pairwise disjoint sets of variables. Notice, however, that for every $t \in H$ the query q_t is equivalent to $\text{CONSTRUCT } t \text{ WHERE } P_t$, and hence satisfies the three properties mentioned above. Finally, define H' and P' as:

$$H' = \{t' \mid t \in H\} \qquad P' = \underset{t \in H}{\text{UNION}}\, P_t.$$

Let $q' = \text{CONSTRUCT } H' \text{ WHERE } P'$. We prove that q and q' are equivalent. Let G be an RDF graph.

- $\Rightarrow$ Let $\mu \in [\![P]\!]_G$ and $t \in H$ such that $\mu(t) \in \text{ans}(q, G)$. By the first property proved above, we know that $\mu(t)$ is in the answer to $\text{CONSTRUCT } t \text{ WHERE } P_t$ over G, which implies that $\mu(t) \in \text{ans}(\text{CONSTRUCT } t' \text{ WHERE } P'_t, G)$. Since P'_t is one of the disjuncts of P' and $t' \in H'$, we have that $\mu(t) \in \text{ans}(q', G)$.
- $\Leftarrow$ Let $\mu \in [\![P']\!]_G$ and $t \in H'$ such that $\mu(t) \in \text{ans}(q', G)$. We know that $\mu \in [\![P'_s]\!]_G$ for some $s' \in H'$. If $\text{var}(t) \neq \emptyset$ then such s' must be t' as P'_t and P'_s do not share variables. In this case, $\mu(t) \in \text{ans}(\text{CONSTRUCT } t' \text{ WHERE } P'_t, G)$,

which implies that $\mu(t) \in \text{ans}(\text{CONSTRUCT } t \text{ WHERE } P_t, G)$. By the second property, this implies $\mu(t) \in \text{ans}(q, G)$. On the other hand, tf t has no variables, then we still know that $\mu \in [\![P'_s]\!]_G$ for some $s' \in H'$. This entails there is a mapping in $[\![P_s]\!]_G$. Hence, there is either a mapping μ' in $[\![P]\!]_G$ or in $[\![P^h]\!]_G$ for some $h \in H$. Since P^h is a renaming of P, in any case there must be a mapping $\mu' \in [\![P]\!]_G$. Finally, as t has no variables, $\mu'(t) = \mu(t) \in [\![P]\!]_G$.

We proved that q is equivalent to $q' = \text{CONSTRUCT } H' \text{ WHERE } P'$. Since P' is a disjunction between weakly-monotone graph patterns, we know that P' is also weakly monotone, which concludes the proof. □

From the previous two lemmas we can finally obtain a syntactic characterization of monotonicity.

THEOREM 11. *The fragment of monotone c-queries is equivalent to the fragment of OPT-free c-queries.*

PROOF. Since OPT-free graph patterns are actually monotone, it immediately follows that OPT-free c-queries are monotone. We proceed with the other direction. Let q be a monotone c-query. From Lemma 10, we can assume without loss of generality that $q = \text{CONSTRUCT } H \text{ WHERE } P$, with P being weakly-monotone. Therefore, from Theorem 9, we know there is an OPT-free graph pattern Q that is equivalent to P in maximal answers. This implies that $\text{NS}(P) \equiv \text{NS}(Q)$, and hence by applying Lemma 9 we obtain

$$\text{CONSTRUCT } H \text{ WHERE } P \equiv \text{CONSTRUCT } H \text{ WHERE NS}(P)$$

$$\equiv \text{CONSTRUCT } H \text{ WHERE NS}(Q) \equiv \text{CONSTRUCT } H \text{ WHERE } Q$$

This concludes the proof as Q is OPT-free. □

This result shows a first characterization of monotonicity. However, we are able to strengthen this result by proving that projection is actually not necessary. To this end, we need to define the SELECT-free version of a graph pattern.

DEFINITION 7.3.2. *Let P be a SPARQL$_{NS}$ graph pattern. The SELECT-free version of P, denoted by P_{sf}, is recursively defined as follows:*

- *If P is a triple pattern, then $P_{\mathsf{sf}} = P$.*
- *If $P = \text{SELECT}\, V \;\text{ WHERE }\; P'$, then P_{sf} is the result of replacing in P'_{sf} every variable in $var(P') \setminus V$ by a fresh variable. Notice that the SELECT is removed.*
- *If P is $(P_1 * P_2)$, where $*$ is one of $\{AND, UNION, FILTER\}$, then $P_{\mathsf{sf}} = (P_{1\mathsf{sf}} * P_{2\mathsf{sf}})$, assuming that the sets of variables $var(P_{1\mathsf{sf}}) \setminus var(P)$ and $var(P_{2\mathsf{sf}}) \setminus var(P)$ are disjoint.*
- *If $P = NS(P')$, then $P_{\mathsf{sf}} = NS(P'_{\mathsf{sf}})$.*
- *If $P = P' \text{ FILTER } R$, then $P = P'_{\mathsf{sf}} \text{ FILTER } R$.*

To show that under CONSTRUCT the operator SELECT is not necessary we prove the following lemma.

LEMMA 11. *Let P be a graph pattern. For every RDF graph G, a mapping μ is in $[\![P]\!]_G$ if and only if there is a mapping $\mu' \in [\![P_{\mathsf{sf}}]\!]_G$ such that $\mu \preceq \mu'$ and $dom(\mu) = dom(\mu') \cap var(P)$.*

PROOF. We proceed by induction. Assume G is an RDF graph and let μ be a mapping.

- If P is a triple pattern the result immediately follows.
- If P is $P_1 \text{ UNION } P_2$, then $\mu \in [\![P]\!]_G$ if and only if $\mu \in [\![P_1]\!]_G \cup [\![P_2]\!]_G$. By hypothesis, this occurs if and only if there is a mapping $\mu' \in [\![P_{1\mathsf{sf}}]\!]_G \cup [\![P_{2\mathsf{sf}}]\!]_G$ such that $\mu \preceq \mu'$. This concludes the proof as $[\![P_{1\mathsf{sf}}]\!]_G \cup [\![P_{2\mathsf{sf}}]\!]_G = [\![P_{\mathsf{sf}}]\!]_G$.
- Let $P = P_1 \text{ AND } P_2$. If $\mu \in [\![P]\!]_G$,then there are two mappings $\mu_1 \in [\![P_1]\!]_G$ and $\mu_2 \in [\![P_2]\!]_G$ such that $\mu = \mu_1 \cup \mu_2$. By hypothesis, this implies there are two mappings $\mu'_1 \in [\![P_{1\mathsf{sf}}]\!]_G$ and $\mu'_2 \in [\![P_{2\mathsf{sf}}]\!]_G$ such that $\mu_1 \preceq \mu'_1$ and $\mu_2 \preceq \mu'_2$. We know that μ_1 and μ_2 only mention variables in var(P), and are compatible. This implies that μ'_1 and μ'_2 are compatible, as they cannot have variables in common

that are not mentioned in var(P). Then, $\mu_1 \cup \mu_2 \in [\![P_{\mathsf{sf}}]\!]_G$, which concludes this direction as $\mu \preceq \mu_1 \cup \mu_2$. The opposite direction is proved by the same argument.

- Let $P = P_1$ OPT P_2. We know that $\mu \in [\![P]\!]_G$ if and only if $\mu \in [\![P_1 \text{ MINUS } P_2]\!]_G$ or $\mu \in [\![P_1 \text{ AND } P_2]\!]_G$. We prove that $\mu \in [\![P_1 \text{ MINUS } P_2]\!]_G$ if and only if $\mu \in [\![P_1 \text{ MINUS } P_{2\mathsf{sf}}]\!]_G$, and that $\mu \in [\![P_1 \text{ AND } P_2]\!]_G$ if and only if $\mu \in [\![P_1 \text{ AND } P_{2\mathsf{sf}}]\!]_G$. The latter case was already proved. By definition, $\mu \in [\![P_1 \text{ MINUS } P_2]\!]_G$ if and only if $\mu \in [\![P_1]\!]_G$ and there is no $\mu' \in [\![P_2]\!]_G$ compatible with μ. By hypothesis this occurs if and only if there is a mapping $\mu_1 \in [\![P_{1\mathsf{sf}}]\!]_G$ such that $\mu \preceq \mu_1$, and there is no mapping $\mu_2 \in [\![P_{2\mathsf{sf}}]\!]_G$ compatible with μ (as μ_2 would be the extension of a mapping in $[\![P_2]\!]_G$ compatible with μ). But these conditions occur if and only if $\mu_1 \in [\![P_{1\mathsf{sf}} \text{ MINUS } P_{2\mathsf{sf}}]\!]_G$. This concludes the proof as $\mu \preceq \mu_1$.
- Let $P = P'$ FILTER R. Then $\mu \in [\![P]\!]_G$ if and only if $\mu \in [\![P']\!]_G$ and $\mu \models R$. By hypothesis, we have that $\mu \in [\![P']\!]_G$ if and only if there is a mapping $\mu' \in [\![P'_{\mathsf{sf}}]\!]_G$ such that $\mu \preceq \mu'$. Moreover, the variables in dom(μ') \ dom(μ) are not mentioned in var(P) and hence they are not mentioned in R. Hence $\mu \models R$ if and only if $\mu' \models R$, concluding the proof.
- If $P =$ NS(P'), then $\mu \in [\![P]\!]_G$ implies μ is a maximal mapping in $[\![P']\!]_G$. By hypothesis, this implies there is a mapping in $[\![P'_{\mathsf{sf}}]\!]_G$ subsuming μ, and hence there must be a mapping in $[\![NS(P'_{\mathsf{sf}})]\!]_G$ subsuming μ. For the opposite direction, assume $\mu'[\![NS(P'_{\mathsf{sf}})]\!]_G$. Then, μ' is a maximal mapping in $[\![P'_{\mathsf{sf}}]\!]_G$. Therefore, there must be a maximal mapping $\mu \in [\![P']\!]_G$ such that $\mu \preceq \mu'$. It is easy to see that if this was not the case then μ' would not have been maximal.
- Let $P =$ SELECT V WHERE P'. Assume $\mu \in [\![P]\!]_G$. Then, there is a mapping $\mu' \in [\![P']\!]_G$ such that $\mu \preceq \mu'$. As P_{sf} results from renaming variables not mentioned in dom(μ) in P', there must be a mapping μ'' that is a renaming of μ' subsuming μ. The opposite direction readily follows from a similar argument.

□

Having this lemma, we can prove that SELECT does not provide more expressive power under CONSTRUCT.

THEOREM 12. *The fragment of* AUF *c-queries is equivalent in expressive power to the fragment of* AUF^{π} *c-queries.*

PROOF. We only need to prove that every AUF^{π} c-query can be transformed into an AUF c-query. Let $q =$ CONSTRUCT H WHERE P be an AUF^{π} c-query. We can assume w.l.o.g that $\text{var}(H) \subseteq \text{var}(P)$. We prove that $q \equiv$ CONSTRUCT H WHERE P_{sf}. Let G be an RDF graph.

- [$\Rightarrow$] Assume $\mu(t) \in \text{ans}(q, G)$, where $\mu \in [\![P]\!]_G$ and $t \in H$. Then, there is a mapping μ' in $[\![P_{\mathsf{sf}}]\!]_G$ such that $\mu \preceq \mu'$. This implies that $\mu'(t)$ is a triple in ans(CONSTRUCT H WHERE P_{sf}, G). As $\mu \preceq \mu'$, we obtain that $\mu'(t) = \mu(t)$, concluding this direction.
- [$\Rightarrow$] Assume $\mu'(t) \in$ ans(CONSTRUCT H WHERE P_{sf}, G), where $\mu' \in [\![P_{\mathsf{sf}}]\!]_G$ and $t \in H$. Then, there is a mapping μ in $[\![P]\!]_G$ such that $\mu \preceq \mu'$ and $\text{dom}(\mu) = \text{dom}(\mu') \cap \text{var}(P)$. Since $\text{var}(t) \subseteq \text{var}(P)$ and $\text{var}(t) \subseteq \text{dom}(\mu')$, we obtain that $\text{var}(t) \subseteq \text{dom}(\mu)$. We conclude that $\mu(t) \in \text{ans}(q, G)$.

$\square$

Finally, we deduce as a corollary what is probably the most important result of this dissertation.

COROLLARY 6. *The fragment of monotone c-queries is equivalent in expressive power to the fragment of* AUF *c-queries.*

This result is somewhat unexpected as it reduces the complete query language of monotone c-queries to a simple fragment in which only basic operations are allowed.

We have presented the advantages of producing RDF graphs as output, and found a clean and simple syntactic characterization of monotonicity under c-queries. Some of these results and further novel properties of c-queries, like the relation between c-queries with blank nodes in templates and data exchange settings, have been published in (Kostylev, Reutter, & Ugarte, 2015). We think this is enough evidence to say that AUF c-queries are a very practical fragment, and should be considered as an important language for querying RDF.

7.4. Well-designed CONSTRUCT queries

In section 4.4 we explained why to apply interpolation it is necessary to allow for infinite RDF graphs. The consequence of including infinite graphs is that the characterizations found for SPARQL fragments, in particular the one for monotone c-queries, are not necessarily complete for finite graphs. Indeed, it might be the case that a c-query q is monotone under finite graphs, but there is no c-query in AUF equivalent to q. Finding such a query in practical applications is not expected (Ajtai & Gurevich, 1987). For example, when proving that well-designed graph patterns are weakly-monotone, no assumption is made about the cardinality of RDF graphs, and therefore it is not hard to see that every well-designed graph pattern is weakly-monotone under arbitrary RDF graphs. From this we can conclude that the characterization presented in Corollary 6 captures all c-queries with well-designed graph patterns. Moreover, since AUF graph patterns are well-designed, the set of AUF c-queries and c-queries with well-designed graph patterns are equivalent in expressive power.

Given that well-designed graph patterns are a well-established and widely adopted fragment of SPARQL, c-queries with well-designed graph patterns is a fragment that could be easily adopted. Hence, a natural question to ask is whether we should include well-designed OPT in our syntactic characterization. This simplifies the query-writing process, but affects the query evaluation as now left outer-joins must be performed. To answer this question, in this section we show that we can effectively transform c-queries with well-designed graph patterns into AUF c-queries. With this transformation there is no need to perform left outer-joins, as queries could be processed before evaluation to remove the occurrences of OPT.

We say that a c-query q is well-designed if the graph pattern of q is well-designed. In order to perform the aforementioned transformation we need to recall some notions and results from (Pérez et al., 2009). Given two SPARQL graph patterns P and P', we say that P' is a direct reduction of P if P' can be obtained from P by replacing a sub pattern of the form $(P_1 \ \text{OPT} \ P_2)$ by P_1. The reflexive and transitive closure of this relation is denoted by $\trianglelefteq$. For every pattern P, and(P) is the result of replacing in P every sub pattern of the form $(P_1 \ \text{OPT} \ P_2)$ by $(P_1 \ \text{AND} \ P_2)$. Given a pattern P and an RDF graph G, a mapping μ is said to be a partial solution to P over G if there is a graph pattern P' such that $P' \trianglelefteq P$ and $\mu \in [\![\text{and}(P')]\!]_G$. Notice that the above definitions are general to all SPARQL graph patterns, including projection.

In (Pérez et al., 2009) the authors proved that given a well-designed graph pattern P, a mapping μ is in $[\![P]\!]_G$ if and only if μ is a maximal partial solution to P over G. As we know the only mappings of interest under c-queries are the maximal ones. Thus, we define a transformation that given a well-designed graph pattern P, produces a graph pattern in AUF that outputs all partial solutions to P. For every pattern P define P_{of} as the result of replacing in P every pattern of the form $(P_1 \ \text{OPT} \ P_2)$ by $(P_1 \ \text{UNION} \ (P_1 \ \text{AND} \ P_2))$.

LEMMA 12. *For every RDF graph G and SPARQL graph pattern P, it is the case that $[\![P_{\mathsf{of}}]\!]_G$ is the set of partial solutions to P over G.*

PROOF. We start by showing that every partial solution to P over G is in $[\![P_{\mathsf{of}}]\!]_G$. Let μ be a partial solution to P over G. We proceed by induction over P.

- If $P = P_1 \ \text{UNION} \ P_2$, then without loss of generality we can assume that μ is a partial solution to P_1 over G. Then μ belongs to $[\![P_{1\mathsf{of}}]\!]_G$, and hence to $[\![P_{1\mathsf{of}} \ \text{UNION} \ P_{2\mathsf{of}}]\!]_G = [\![P_{\mathsf{of}}]\!]_G$.

- If $P = P_1$ AND P_2, then there are two mappings μ_1 and μ_2, with $\mu = \mu_1 \cup \mu_2$, which are partial solutions to P_1 and to P_2 over G, respectively. By hypothesis, $\mu_1 \in [\![P_{1\mathsf{of}}]\!]_G$ and $\mu_2 \in [\![P_{2\mathsf{of}}]\!]_G$. Hence, $\mu \in [\![P_{1\mathsf{of}}$ AND $P_{2\mathsf{of}}]\!]_G = [\![P_{\mathsf{of}}]\!]_G$.
- If $P = P_1$ FILTER R, then μ is a partial solution to P_1 which satisfies R. Hence, μ belongs to $[\![P_{1\mathsf{of}}$ FILTER $R]\!]_G = [\![P_{\mathsf{of}}]\!]_G$.
- If $P =$ SELECT V WHERE P', then $\mu = \mu'_{|V}$ for some $\mu' \in [\![P']\!]_G$. Therefore, μ' is in $[\![P'_{\mathsf{of}}]\!]_G$ and hence in $[\![$SELECT V WHERE $P'_{\mathsf{of}}]\!]_G$, which is equivalent to $[\![P_{\mathsf{of}}]\!]_G$.
- If $P = P_1$ OPT P_2, either μ is a partial solution to P_1 over G or μ is a partial solution to P_1 AND P_2 over G. Then, we know by hypothesis that $\mu \in [\![P_{1\mathsf{of}}]\!]_G$ or $\mu \in [\![P_{1\mathsf{of}}$ AND $P_{2\mathsf{of}}]\!]_G$. In either case we have $\mu \in [\![P_{1\mathsf{of}}$ UNION $(P_{1\mathsf{of}}$ AND $P_{2\mathsf{of}})]\!]_G$, which is the definition of $[\![P_{\mathsf{of}}]\!]_G$.

Next we prove that every mapping $\mu \in [\![P_{\mathsf{of}}]\!]_G$ is a partial solution to P over G. Again, we proceed by induction on P. If P is a triple, UNION -, AND -, SELECT-, or FILTER - pattern, the result readily follows by the induction hypothesis as in the previous case. If $P = (P_1$ OPT $P_2)$, then we know that $P_{\mathsf{of}} = (P_{1\mathsf{of}}$ UNION $(P_{1\mathsf{of}}$ AND $P_{2\mathsf{of}}))$. Therefore, μ may either be in $[\![P_{1\mathsf{of}}]\!]_G$ or in $[\![(P_{1\mathsf{of}}$ AND $P_{2\mathsf{of}})]\!]_G$. Then μ is a partial result to P_1 over G or to $(P_1$ AND $P_2)$ over G. In both cases, μ is a partial solution to $(P_1$ OPT $P_2)$ over G. $\square$

With this result we are ready to state the procedure that transform a well-designed c-query into an AUF c-query.

THEOREM 13. *There is an algorithm that, given a well-designed c-query q, generates an* AUF *c-query q' that is equivalent to q. Moreover, this transformation takes at most exponential time in the number of occurrences of the OPT operator in q.*

PROOF. Let $q =$ CONSTRUCT H WHERE P be a well-designed c-query. Let q_{of}, the OPT-free version of q, be defined as CONSTRUCT H WHERE P_{of}. We show that $q \equiv q_{\mathsf{of}}$.

- [$\Rightarrow$] Let G be an RDF graph and let $\mu(t)$ be a triple in $\text{ans}(q, G)$, where $\mu \in [\![P]\!]_G$ and $t \in H$. Since P is well-designed, we know that μ is a maximal partial solution to P over G. By Lemma 12, this entails that $\mu \in [\![P_{\mathsf{of}}]\!]_G$ and therefore $\mu(t) \in \text{ans}(q_{\mathsf{of}}, G)$.
- [$\Leftarrow$] Let G be an RDF graph and let $\mu(t)$ be a triple in $\text{ans}(q_{\mathsf{of}}, G)$, where $\mu \in [\![P_{\mathsf{of}}]\!]_G$ and $t \in H$. By Lemma 12, μ is a partial solution to P over G. Then, there is a mapping μ' that is a maximal partial solution to P over G extending μ. This implies that $\mu'(t) \in$ CONSTRUCT H WHERE $NS(P_{\mathsf{of}})$, as $\mu' \in [\![NS(P_{\mathsf{of}})]\!]_G$. From Lemma 9, we obtain $\mu'(t) \in$ CONSTRUCT H WHERE $NS(P_{\mathsf{of}})$. This concludes the proof as $\mu'(t) = \mu(t)$.

Finally, it is easy to see that constructing the OPT-free version of a c-query can take at most exponential time, as when removing one occurrence of OPT the remaining nested occurrences can at most double. □

Contrary to the characterizations relying on interpolation, the previous theorem provides an effective construction of an equivalent c-query. It is important to mention that although there is an exponential time upper bound, this transformation is suitable for practical applications as SPARQL queries generally contain a small number of OPT occurrences.

As already mentioned, the equivalence between well-designed c-queries and AUF c-queries does not depend on a condition over arbitrary models (as weak-monotonicity in the case of graph patterns). This has an interesting consequence. On one hand, we have that AUF c-queries are equivalent to all queries that are monotone under arbitrary models. On the other hand, we know that AUF c-queries are equivalent to all well-designed c-queries, even under finite models. This means that well-designed c-queries are precisely equivalent to the set of c-queries that are monotone under arbitrary models.

Recall that in Chapter 3, we exposed the graph pattern

$$P = (?X, a, b) \text{ OPT } ((?X, c, ?Y) \text{ UNION } (?X, d, ?Z))$$

as a weakly-monotone graph pattern that cannot be expressed as a well-designed graph pattern. According to the previous result, a monotone c-query q cannot be expressed as a well-designed c-query if and only if q is monotone under finite models, but not under arbitrary models. Therefore, the c-query q = CONSTRUCT H WHERE P must be equivalent to a well-designed c-query, and actually to an AUF c-query. It is not hard to realize that

$$\begin{aligned} q \equiv \text{CONSTRUCT } H \text{ WHERE } [&(?X, a, b) \text{ UNION} \\ &((?X, a, b) \text{ AND } (?X, c, ?Y)) \text{ UNION } ((?X, a, b) \text{ AND } (?X, d, ?Z))]. \end{aligned}$$

In order to construct a c-query that is monotone under finite graphs and is not equivalent to any c-query in AUF, a much more complex construction is needed. For example, we could create a c-query representing that the evaluated RDF graph corresponds to a linear order without a last element. It is not the goal of this dissertation to show such an example, but this illustrates that for practical applications, the set of AUF c-queries characterizes monotonicity.

At this point of the dissertation we have introduced several characterizations of various fragments of SPARQL graph patterns and c-queries. Moreover, for the case of c-queries we showed a simple algorithm that transforms any well-designed c-query into an AUF c-query. In the next chapter, we address the performance problems of the Semantic Web mentioned in the introduction. We study practicality of the found languages from the point of view of the evaluation complexity.

8. COMPUTATIONAL COMPLEXITY

In the previous two chapters, we introduced a new operator and several syntactic fragments for characterizing notions related to open-world semantics. It is natural to ask whether these fragments are practical for real-world applications, in particular compared to previously established fragments of SPARQL. To answer this question, we study the combined computational complexity (Vardi, 1982) associated to the evaluation problem. This problem consists on deciding, given an RDF graph G, a graph pattern P and a mapping μ, whether μ belongs to $[\![P]\!]_G$.

Our complexity results are built upon several studies of the complexity of evaluating SPARQL graph patterns (Pichler & Skritek, 2014; Pérez et al., 2009; Schmidt et al., 2010; Arenas & Pérez, 2011; Letelier, Pérez, Pichler, & Skritek, 2012). The fundamental ideas rely on the fact that the evaluation problem is NP-complete for OPT-free graph patterns in SPARQL (Schmidt et al., 2010), and is CO-NP-complete for well-designed graph patterns (Pérez et al., 2006a). Formally, the evaluation problem for a fragment $\mathcal{F}$ is defined as the following language

$$\{(G, P, \mu) \mid G \text{ is an RDF graph, } P \text{ is a graph pattern in } \mathcal{F}, \text{ and } \mu \in [\![P]\!]_G\}.$$

8.1. Complexity Classes

We use complexity classes that might not be familiar to the reader, and hence we briefly recall their definition. In particular, we present the Boolean Hierarchy and the complexity class $\mathrm{P}_{\parallel}^{\mathrm{NP}}$.

The Boolean Hierarchy is an infinite family of complexity classes based on boolean combinations of languages in NP (Wechsung, 1985). The most famous class in this hierarchy is DP, which consists of all languages that can be expressed as $L_1 \cap L_2$, where $L_1 \in$ NP and $L_2 \in$ CO-NP. The levels of the boolean hierarchy are denoted by $\{BH_i\}_{i \in \mathbb{N}}$, and are recursively defined as follows:

- BH_1 is the complexity class NP.
- BH_{2k} consists of all languages that can be expressed as $L_1 \cap L_2$, where $L_1 \in BH_{2k-1}$ and $L_2 \in$ co-NP.
- BH_{2k+1} consists of all languages that can be expressed as $L_1 \cup L_2$, where $L_1 \in BH_{2k}$ and $L_2 \in$ NP.

The complexity class $\mathrm{P}_{\|}^{\mathrm{NP}}$ (Hemachandra, 1989) contains all problems that can be solved in polynomial time by a Turing machine that can make a polynomial (in terms of the input's length) amount of queries, in parallel, to an NP oracle. The fact that the access to the oracle is in parallel basically prevents the queries to depend on previous oracle answers. The class $\mathrm{P}_{\|}^{\mathrm{NP}}$ was proved to be equal to $\Delta_2^P[\log n]$, the complexity class of all problems that can be solved in polynomial time by a Turing machine that can make $O(\log n)$ queries to an NP oracle, not necessarily in parallel (Buss & Hay, 1991).

8.2. Complexity of simple patterns

We start by studying the complexity of simple graph patterns, which will serve as a yardstick for studying the complexity of the more general fragments. To establish the complexity of simple graph patterns, we need to take a close look into the proof of NP-completeness of evaluating AUF, originally presented in (Pérez et al., 2009).

LEMMA 13. *[(Pérez et al., 2009), Theorem 3.2] There is a polynomial-time algorithm that, given a propositional formula φ, generates a mapping μ_φ, a graph pattern P_φ in* AUF *and an RDF graph G_φ, such that:*

(1) *$dom(\mu_\varphi) = var(P_\varphi)$ and $\mathbf{I}(P_\varphi) = \mathbf{I}(G_\varphi)$.*
(2) *Every triple pattern in P_φ mentions variables and IRIs.*
(3) *If φ is satisfiable, then $[\![P]\!]_{G_\varphi} = \{\mu_\varphi\}$.*
(4) *If φ is unsatisfiable, then $[\![P]\!]_{G_\varphi} = \emptyset$.*

LEMMA 14. *Let G_1 and G_2 be two RDF graphs such that $\mathbf{I}(G_1) \cap \mathbf{I}(G_2) = \emptyset$, and let P be a graph pattern in SPARQL$_{NS}$. If P is free from variable-only triple patterns and $\mathbf{I}(P) \subseteq \mathbf{I}(G_1)$, then $[\![P]\!]_{G_1 \cup G_2} = [\![P]\!]_{G_1}$.*

PROOF. Proceed by induction over the structure of P. If P is a triple pattern, then it must mention some IRI in that is in $\mathbf{I}(G_1) \setminus \mathbf{I}(G_2)$. Hence, P can only match triples in G_1. It follows that $[\![P]\!]_{G_1 \cup G_2} = [\![P]\!]_{G_1}$. The remaining cases are proven directly from the inductive definition of SPARQL:

- If $P = P_1$ AND P_2, we have $[\![P]\!]_{G_1 \cup G_2} = [\![P_1]\!]_{G_1 \cup G_2} \bowtie [\![P_2]\!]_{G_1 \cup G_2}$. By induction hypothesis this is the same as $[\![P_1]\!]_{G_1} \bowtie [\![P_2]\!]_{G_1} = [\![P]\!]_{G_1}$.
- If $P = P_1$ UNION P_2, we have $[\![P]\!]_{G_1 \cup G_2} = [\![P_1]\!]_{G_1 \cup G_2} \cup [\![P_2]\!]_{G_1 \cup G_2}$. By hypothesis this is the same as $[\![P_1]\!]_{G_1} \cup [\![P_2]\!]_{G_1} = [\![P]\!]_{G_1}$.
- If $P = P_1$ OPT P_2, we have $[\![P]\!]_{G_1 \cup G_2} = [\![P_1]\!]_{G_1 \cup G_2} ⟕ [\![P_2]\!]_{G_1 \cup G_2}$. From the hypothesis this is the same as $[\![P_1]\!]_{G_1} ⟕ [\![P_2]\!]_{G_1} = [\![P]\!]_{G_1}$.
- If $P = P'$ FILTER R, we have $[\![P]\!]_{G_1 \cup G_2} = \{\mu \in [\![P']\!]_{G_1 \cup G_2} \mid \mu \models R\}$. By induction hypothesis this is the same as $\{\mu \in [\![P']\!]_{G_1} \mid \mu \models R\} = [\![P]\!]_{G_1}$.
- If $P = \text{NS}(P')$, the result immediately follows as $[\![P']\!]_{G_1 \cup G_2} = [\![P']\!]_{G_1}$.

□

Now we are ready to state the complexity of simple graph patterns.

THEOREM 14. *The evaluation problem for simple graph patterns is* DP*-complete.*

PROOF. According to the definition, the evaluation problem for simple graph patterns corresponds to the language of all triples (G, P, μ) such that $\mu \in [\![P]\!]_G$, where $P = \text{NS}(P')$ is a simple pattern. This language is in DP since it can be expressed as:

$$\{(G, P, \mu) \mid\ P = \text{NS}(P') \text{ is a simple pattern and } \mu \in [\![P']\!]_G\} \cap$$
$$\{(G, P, \mu) \mid P = \text{NS}(P') \text{ is a s.p. and there is no } \mu' \in [\![P']\!]_G \text{ s.t. } \mu \prec \mu'\}.$$

The first language is clearly in NP, as it is only necessary to solve the evaluation problem for a pattern in AUF^{π}. The second language is in co-NP as its complement consists of the triples $(G, \mathrm{NS}(P), \mu)$ where $P = NS(P')$ (which is polynomially verifiable) and there is a mapping $\mu' \in [\![P']\!]_G$ such that $\mu \prec \mu'$. This is also in NP as μ' can be guessed and P' is in AUF^{π}.

We show that the evaluation problem for simple patterns is DP-hard. We provide a reduction from the well-known DP-complete problem SAT-UNSAT (Papadimitriou & Yannakakis, 1982). This is the problem of deciding, given a pair of propositional formulas (φ, ψ), whether φ is satisfiable and ψ is unsatisfiable.

Let (φ, ψ) be a pair of propositional formulas. Let μ_φ, P_φ, G_φ and μ_ψ, P_ψ, G_ψ be the elements provided by Lemma 13 corresponding to φ and ψ, respectively. By renaming variables and IRIs, we can assume w.l.o.g. that the IRIs and variables mentioned in μ_φ, P_φ, G_φ are disjoint from those mentioned in μ_ψ, P_ψ, G_ψ. Next we prove that μ_φ belongs to the answer to

$$P = \mathrm{NS}(P_\varphi \ \text{UNION}\ (P_\varphi \ \text{AND}\ \ P_\psi))$$

over the RDF graph $G = G_\varphi \cup G_\psi$ if and only if φ is satisfiable and ψ is unsatisfiable. Notice that by Lemma 14 we have that $\mu_\varphi \in [\![P_\varphi]\!]_G$ if and only if $\mu_\varphi \in [\![P_\varphi]\!]_{G_\varphi}$, and that $\mu_\psi \in [\![P_\psi]\!]_G$ if and only if $\mu_\varphi \in [\![P_\psi]\!]_{G_\psi}$.

($\Rightarrow$) Suppose, for the sake of contradiction, that

$$\mu_\varphi \in [\![\mathrm{NS}(P_\varphi \ \text{UNION}\ (P_\varphi \ \text{AND}\ \ P_\psi))]\!]_G,$$

and that φ is unsatisfiable or ψ is satisfiable. We analyze these cases separately.

- If φ is not satisfiable, then we know by Lemma 13 that $[\![P_\varphi]\!]_G = \emptyset$, which implies that $[\![\mathrm{NS}(P_\varphi \ \text{UNION}\ (P_\varphi \ \text{AND}\ \ P_\psi))]\!]_G = \emptyset$.
- If ψ is satisfiable, then we have by Lemma 13 that $\mu_\psi \in [\![P_\psi]\!]_G$. Since $\mathrm{var}(P_\varphi) \cap \mathrm{var}(P_\psi) = \emptyset$ and $[\![P_\psi]\!]_G \neq \emptyset$, every mapping in $[\![P_\varphi]\!]_G$ is subsumed

by some mapping in $[\![P_\varphi \text{ AND } P_\psi]\!]_G$. Hence, we obtain that

$$[\![\text{NS}(P_\varphi \text{ AND } P_\psi)]\!]_G \equiv [\![\text{NS}(P_\varphi \text{ UNION } (P_\varphi \text{ AND } P_\psi))]\!]_G.$$

We know by Lemma 13 that the empty mapping does not belong to $[\![P_\psi]\!]_G$, and therefore every mapping in $[\![P_\varphi \text{ AND } P_\psi]\!]_G$ mentions some variable in $\text{var}(P_\psi)$. As $\text{var}(\mu_\varphi) \cap \text{var}(P_\psi) = \emptyset$, we conclude that $\mu_\varphi \notin [\![P_\varphi \text{ AND } P_\psi]\!]_G$. Thus, μ_φ is not in $[\![\text{NS}(P_\varphi \text{ UNION } (P_\varphi \text{ AND } P_\psi))]\!]_G$, which contradicts our initial assumption.

($\Leftarrow$) Assume φ is satisfiable and ψ is unsatisfiable. By Lemma 13, this implies that $[\![P_\psi]\!]_G = \emptyset$. Hence, in this case we have that $[\![\text{NS}(P_\varphi \text{ UNION } (P_\varphi \text{ AND } P_\psi))]\!]_G$ is the same as $[\![\text{NS}(P_\varphi)]\!]_G$. From Lemma 13, we have that $[\![P_\varphi]\!]_G = \{\mu_\varphi\}$ and, therefore, $\mu_\varphi \in [\![\text{NS}(P_\varphi)]\!]_G$, concluding the proof.

□

It is interesting to notice that the evaluation of simple patterns is already higher than that of well-designed patterns (CO-NP-complete). Next we proceed to study the evaluation complexity of ns-patterns.

8.3. Complexity of ns-patterns

We start by showing that if the amount of disjuncts is bounded by a fixed number k, then the evaluation problem becomes complete for the class BH_{2k}. We first need to prove the following lemma.

LEMMA 15. *Let $n \in \mathbb{N}$, and for every $i \in \{1, \ldots, n\}$ let μ_i, G_i and P_i be a mapping, an RDF graph and a graph pattern, respectively. If the following conditions hold*

- *for every $i, j \in \{1, \ldots, n\}$ with $i \neq j$, the variables and iris mentioned in (μ_i, P_i, G_i) are disjoint from the variables and iris mentioned in (μ_j, P_j, G_j);*

- *for every $i \in \{1, \dots, n\}$, it is the case that P_i is a simple pattern which does not mention variable-only triple patterns,*

then there is a mapping μ, an ns-pattern P and an RDF graph G such that $\mu \in [\![P]\!]_G$ if and only if $\mu_i \in [\![P_i]\!]_{G_i}$ for some $i \in \{1, \dots, n\}$. Moreover, μ, P and G can be computed in polynomial time.

PROOF. First, define the mapping μ as $\mu_1 \cup \mu_2 \cup \cdots \cup \mu_n$. This mapping is correctly defined since $\text{var}(\mu_i) \cap \text{var}(\mu_j) = \emptyset$ for every $i, j \in \{1, \dots, n\}$ with $i \neq j$.

Now define the graph G as

$$\left(\bigcup_{i \in \{1,\dots,n\}} G_i \right) \cup \left(\bigcup_{?X \in \text{dom}(\mu)} (\mu(?X), c^{?X}, d^{?X}) \right)$$

where $c^{?X}$ and $d^{?X}$ are distinct fresh IRIs for every $?X \in \text{dom}(\mu)$. Adding the new IRIs and their corresponding triples allows us to trivially match the graph to include the assignment $?X \to \mu(?X)$ in any mapping not mentioning $?X$. Based on this intuition we proceed to create the ns-pattern P. Let $i \in \{1, \dots, n\}$ and assume that $\text{dom}(\mu) \setminus \text{dom}(\mu_i) = \{?X_1, \dots, ?X_\ell\}$. Assuming that $P_i = NS(Q_i)$, define the pattern P'_i as

$$P'_i = \text{NS}\Big(Q_i \text{ AND } (?X_1, c^{?X_1}, d^{?X_1}) \text{ AND } \cdots \text{ AND } (?X_\ell, c^{?X_\ell}, d^{?X_\ell})\Big).$$

Finally, define the graph pattern P by

$$P = P'_1 \text{ UNION } P'_2 \text{ UNION } \cdots \text{ UNION } P'_n$$

It is clear that the above elements μ, P and G can be computed in polynomial time. Notice that if $\mu_i \in [\![P_i]\!]_{G_i}$ then μ_i will appear in the answer to Q_i over G, as $G_i \subseteq G$ and Q_i is monotone. Moreover, for every $?X \in \text{dom}(\mu) \setminus \text{dom}(\mu_i)$ the triple pattern $(?X, c^{?X}, d^{?X})$ will trivially match the RDF triple $(\mu(?X), c^{?X}, d^{?X})$.

Now that we have defined μ, P and G, we formally prove that μ is in $[\![P]\!]_G$ if and only if $\mu_i \in [\![P_i]\!]_{G_i}$ for some $i \in \{1, \dots, n\}$. Since $P = P'_1$ UNION P'_2 UNION $\cdots$ UNION P'_n, we know that $\mu \in [\![P]\!]_G$ if and only if $\mu \in [\![P'_i]\!]_G$ for some $i \in \{1, \ldots, n\}$. Thus, it is sufficient to show that for each $i \in \{1, \ldots, n\}$ it is the case that

$$\mu_i \in [\![P_i]\!]_{G_i} \text{ if and only if } \mu \in [\![P'_i]\!]_G \tag{8.1}$$

Let $i \in \{1, \ldots, n\}$. Define the mapping μ_{-i} as μ restricted to $\text{dom}(\mu) \setminus \text{dom}(\mu_i)$. Assume $\text{dom}(\mu_{-i}) = \{?X_1, \ldots, ?X_\ell\}$. We have

$$P'_i = \text{NS}\Big(Q_i \text{ AND } (?X_1, c^{?X_1}, d^{?X_1}) \text{ AND } \cdots \text{ AND } (?X_\ell, c^{?X_\ell}, d^{?X_\ell})\Big).$$

Since G contains every triple of the form $(\mu(?X), c^{?X}, d^{?X})$, and the IRIs $c^{?X}$ and $d^{?X}$ are not mentioned anywhere else in G, we know that

$$[\![(?X_1, c^{?X_1}, d^{?X_1}) \text{ AND } \cdots \text{ AND } (?X_\ell, c^{?X_\ell}, d^{?X_\ell})]\!]_G = \{\mu_{-i}\}. \tag{8.2}$$

We make use of this fact to prove both directions of (8.1).

$\Rightarrow$) Assume $\mu_i \in [\![P_i]\!]_{G_i}$. By semantics of NS, it is the case that $\mu_i \in [\![Q_i]\!]_{G_i}$. Since Q_i is monotone and $G_i \subseteq G$, we have $\mu_i \in [\![Q_i]\!]_G$. As μ_i and μ_{-i} are compatible, by equation (8.2) we obtain that

$$\mu_i \cup \mu_{-i} \in [\![Q_i \text{ AND } (?X_1, c^{?X_1}, d^{?X_1}) \text{ AND } \cdots \text{ AND } (?X_\ell, c^{?X_\ell}, d^{?X_\ell})]\!]_G.$$

Finally, as $\mu_i \cup \mu_{-i} = \mu$ and $\text{dom}(\mu) = \text{var}(P'_i)$, we have $\mu \in [\![P'_i]\!]_G$.

$\Leftarrow$) Assume $\mu \in [\![P'_i]\!]_G$. By the semantics of the NS operator, we know that $\mu \in [\![Q_i \text{ AND } (?X_1, c^{?X_1}, d^{?X_1}) \text{ AND } \cdots \text{ AND } (?X_\ell, c^{?X_\ell}, d^{?X_\ell})]\!]_G$. Provided that $[\![(?X_1, c^{?X_1}, d^{?X_1}) \text{ AND } \cdots \text{ AND } (?X_\ell, c^{?X_\ell}, d^{?X_\ell})]\!]_G = \{\mu_{-i}\}$, we have that $[\![Q_i]\!]_G$ must contain a mapping subsuming μ_i. This mapping must be exactly μ_i, as $\text{dom}(\mu_i) = \text{var}(Q_i)$. We conclude that $\mu_i \in [\![\text{NS}(Q_i)]\!]_G = [\![P_i]\!]_G$. From Lemma 14 we know that $[\![P_i]\!]_G = [\![P_i]\!]_{G_i}$, concluding the proof.

$\square$

Now are ready to state the evaluation complexity of ns-patterns with a bounded amount of disjuncts.

THEOREM 15. *Let $k > 0$ be a fixed number. The evaluation problem for ns-patterns with at most k disjuncts is BH_{2k}-complete.*

PROOF. To prove containment we proceed by induction. For $k = 1$, we have the evaluation problem for simple patterns, which is complete for DP $= BH_2$. For every $k > 1$ let $\textsc{Eval}_k$ be the language defined as

$$\textsc{Eval}_k = \{(\mu, P_1 \text{ UNION } \cdots \text{ UNION } P_j, G) \mid j \leq k, \\ P_i \text{ is a simple pattern, and } \mu \in [\![P]\!]_G\}.$$

By induction hypothesis assume $\textsc{Eval}_k$ is in BH_{2k}. Now, we want to show that the problem

$$\textsc{Eval}_{k+1} = \{(\mu, P_1 \text{ UNION } \cdots \text{ UNION } P_j, G) \mid j \leq k+1, \\ P \text{ is an ns-pattern, and } \mu \in [\![P]\!]_G\}$$

is in $BH_{2(k+1)}$. It is easy to see that $\textsc{Eval}_{k+1}$ can be divided into the next two languages (each P_i is a simple pattern).

$$L_1 = \{(\mu, P_1 \text{ UNION } \cdots \text{ UNION } P_j, G) \mid j \leq k+1, \\ \text{and } \mu \in [\![P_i]\!]_G \text{ for some } i \in [1..k]\}$$

$$L_2 = \{(\mu, P_1 \text{ UNION } \cdots \text{ UNION } P_{k+1}, G) \mid \mu \in [\![P_{k+1}]\!]_G\}$$

We have $\textsc{Eval}_{k+1} = L_1 \cup L_2$. Notice that since $\textsc{Eval}_k$ is in BH_{2k}, it is trivial to prove that $L_1 \in BH_{2k}$. Moreover, since $\textsc{Eval}_1$ (evaluation of simple patterns) is in DP, it is trivial to show that L_2 is also in DP. Hence, $\textsc{Eval}_{k+1}$ is the union between a problem in BH_{2k} and a problem in DP. We know from (Wagner, 1987) that such a union belongs to BH_{2k+2}, which concludes the containment proof.

Let $k > 0$. To prove that EVAL_k is BH_{2k}-hard, we make a reduction from the problem of knowing if a graph has chromatic number in the set $M_k = \{6k+1, 6k+3 \ldots, 8k-1\}$. This problem is known as EXACT-M_k-COLORABILITY and is proved to be BH_{2k}-complete in (Riege & Rothe, 2006).

We will create a function that takes a graph H as input and generates an RDF graph G, a mapping μ and a pattern $P = P_1$ UNION $\cdots$ UNION P_k, such that the chromatic number of G is in M_k if and only if $\mu \in [\![P]\!]_G$. Let H be a graph. Denote by $\{m_1, \ldots, m_k\}$ the elements in M_k. As $k > 0$ we know that every element in M_k is greater than 3. Hence, the problem of knowing if a graph has chromatic number m is DP-complete for every m in M_k (Riege & Rothe, 2006). Since the evaluation problem for simple patterns is DP-complete, for every $i \in \{1, \ldots, k\}$ we can generate in polynomial time an RDF graph G_i, a mapping μ_i and a simple pattern P_i, such that $\mu_i \in [\![P_i]\!]_{G_i}$ if and only if H has chromatic number m_i. Moreover, we assume w.l.o.g. that for $i \neq j$, the variables and IRIs mentioned in μ_i, G_i and P_i are disjoint from those mentioned in μ_j, G_j and P_j. Hence, by lemma 15, we can construct in polynomial time a mapping μ, an ns-pattern P with k disjuncts, and an RDF graph G such that $\mu \in [\![P]\!]_G$ if and only if $\mu_i \in [\![P_i]\!]_{G_i}$ for some $i \in \{1, \ldots, k\}$. But as mentioned before, this occurs if and only if H has chromatic number in M_k, and hence the triple (μ, P, G) is the output of our reduction, which concludes the proof. □

Now we proceed to study the complexity of evaluating the full fragment of ns-patterns.

THEOREM 16. EVAL *is* $\text{P}_{\|}^{\text{NP}}$*-complete for ns-patterns.*

PROOF. We prove that the problem belongs to $\text{P}_{\|}^{\text{NP}}$ by providing a straightforward polynomial-time algorithm that calls an NP oracle in parallel a linear number of times. Let $P = P_1$ UNION P_2 UNION $\cdots$ UNION P_n be a graph pattern where every P_i $(1 \leq i \leq n)$ is a simple pattern. Let G be an RDF graph and μ be a mapping. Since for every i the problem of deciding if $\mu \in [\![P_i]\!]_G$ belongs to DP, it can be solved by two parallel calls to an NP oracle. Thus, by making $2n$ calls in parallel to the NP oracle one can decide whether μ belongs to $[\![P_i]\!]_G$ for some $i \in \{1, \ldots, n\}$, and hence decide whether μ belongs to $[\![P]\!]_G$.

Now we prove the problem is $\mathrm{P}_{\|}^{\mathrm{NP}}$-hard by providing a reduction from the problem MAX-ODD-SAT. This is the problem of deciding, given a propositional formula φ, whether the truth-assignment that assigns true to the largest number of variables while satisfying φ, assigns true to an odd number of variables. This problem is shown to be $\mathrm{P}_{\|}^{\mathrm{NP}}$-complete in (Spakowski, 2005).

Let φ be a propositional formula with m variables. We can assume without loss of generality that m is even (if not, consider the formula $\varphi \wedge \neg r$ for a fresh variable r). We want to create an ns-pattern P, an RDF graph G, and a mapping μ such that μ belongs to $[\![P]\!]_G$ if and only if φ belongs to MAX-ODD-SAT. It is easy to see that given a number k between 1 and m, the problem of deciding whether there is a truth assignment that satisfies φ and assigns true to at least k variables is in NP. Thus, by Cook's theorem we can create a propositional formula φ_k such that φ_k is satisfiable if and only if there is a truth assignment that satisfies φ and assigns true to at least k variables. Hence, φ belongs to MAX-ODD-SAT if and only if $(\varphi_k, \varphi_{k+1})$ belongs to SAT-UNSAT for some odd k between 1 and $m-1$. By Theorem 14, for every such k we can create a simple pattern P_k, a mapping μ_k and an RDF graph G_k such that μ_k belongs to $[\![P_k]\!]_{G_k}$ if and only if $(\varphi_k, \varphi_{k+1}) \in$ SAT-UNSAT. We can assume without loss of generality that for every $j, k \in \{1, 3, \ldots, m-1\}$ with $j \neq k$, it is the case that $(\mathrm{dom}(\mu_j) \cup \mathrm{var}(P_j)) \cap (\mathrm{dom}(\mu_k) \cup \mathrm{var}(P_k)) = \emptyset$ and $(\mathrm{range}(\mu_j) \cup \mathbf{I}(P_j) \cup \mathbf{I}(G_j)) \cap (\mathrm{range}(\mu_k) \cup \mathbf{I}(P_k) \cup \mathbf{I}(G_k)) = \emptyset$. Moreover, from the construction on Theorem 14, we know that every pattern P_i $(i \in \{1, 3, \ldots, m-1\})$ begins with the NS operator. Hence, by Lemma 15, we can construct in polynomial time a mapping μ, an ns-pattern P, and an RDF graph G such that $\mu \in [\![P]\!]_G$ if and only if $\mu_i \in [\![P_i]\!]_{G_i}$ for some $i \in \{1, 3, \ldots, m_1\}$.

Next we define a mapping μ, an ns-pattern P and an RDF graph G such that $\mu \in [\![P]\!]_G$ if and only if there is an $i \in \{1, 3, \ldots, m-1\}$ for which $\mu_i \in [\![P_i]\!]_{G_i}$ (recall this occurs if and only if φ belongs to MAX-ODD-SAT). Intuitively, we must simulate a *disjoint union* of mappings, patterns and RDF graphs. The mapping μ is defined as $\mu_1 \cup \mu_3 \cup \cdots \cup \mu_{m-1}$. This mapping is correctly defined since $\mathrm{var}(\mu_j) \cap \mathrm{var}(\mu_k) = \emptyset$ for every $j, k \in \{1, 3, \ldots, m-1\}$ with $j \neq k$. But as mentioned before, this occurs if and only if φ belongs to MAX-ODD-SAT. □

This concludes the complexity study over graph patterns defined with the NS operator. It is interesting to mention that, although the evaluation problem for well-designed graph patterns is CO-NP-complete, well-designed graph patterns do not allow for projection. If projection is allowed only on top, the evaluation of well-designed graph patterns already increases to Σ_2^p-coomplete (Letelier et al., 2013). This complexity is higher than that of ns-patterns, and then it would be interesting to compare the expressive power of this language to that of ns-patterns. Notice also that ns-patterns do allow for projection, but the patterns inside SELECT operator come from AUF. Therefore, with ns-patterns one cannot obtain optional information and then project over some variables. It can be easily shown that allowing projection on top of ns-patterns also produces a Σ_2^p-coomplete language. This is also an interesting language to study.

8.4. Complexity of c-queries

Finally, we state the evaluation complexity for the fragment of AUF c-queries. As mentioned before, this is the most important fragment found in this dissertation, as it correctly captures the set of c-queries that conform to the open-world assumption. Establishing the combined complexity of AUF c-queries is straightforward. We rely on the fact that the evaluation problem for AUF graph patterns is NP-complete.

The evaluation problem for a class $\mathcal{F}$ of c-queries is defined as finding the computational complexity for the following language:

$$\{(G, q, t) \mid G \text{ is an RDF graph, } q \text{ is a c-query in } \mathcal{F}\text{, and } t \in \text{ans}(q, G)\}.$$

THEOREM 17. *The evaluation problem for the* AUF *c-queries is* NP*-complete.*

PROOF. We first show this problem is in NP. Assume G is an RDF graph, t is a triple, and $q = \text{CONSTRUCT } H \text{ WHERE } P$ is an AUF c-query. To prove that t is in $\text{ans}(q, G)$ is suffices to guess a triple $s \in H$ and a mapping $\mu \in [\![P]\!]_G$ such that $t = \mu(s)$. This can be done in NP as $P \in \mathsf{AUF}$.

Now we prove that this problem is NP-hard. We reduce SAT to evaluation of AUF c-queries. Let φ be a propositional formula. Let μ_φ, P_φ and G_φ be the elements provided by Lemma 13. Recall that if φ is satisfiable then $[\![P_\varphi]\!]_{G_\varphi} = \mu_\varphi$, and otherwise $[\![P_\varphi]\!]_{G_\varphi} = \emptyset$. Let $?X$ be a variable in $\text{dom}(\mu_\varphi)$. We prove that the triple $t = (\mu(?X), \mu(?X), \mu(?X))$ is in the answer to

$$q = \text{CONSTRUCT } \{(?X, ?X, ?X)\} \text{ WHERE } P_\varphi$$

over G_φ if and only if φ is satisfiable.

- [$\Rightarrow$] Assume $t \in \text{ans}(q, G_\varphi)$. Then, $[\![P_\varphi]\!]_{G_\varphi}$ is not empty, which occurs if and only if φ is satisfiable.
- [$\Leftarrow$] Assume φ is satisfiable. Then $\mu \in [\![P_\varphi]\!]_{G_\varphi}$, which immediately implies that t is in $\text{ans}(q, G_\varphi)$.

□

This result is interesting as it establishes that the language of AUF c-queries is not only correct in terms of expressive power, but is also a language with low evaluation complexity. In particular, this complexity is much lower than those for general of c-queries and SPARQL graph patterns (PSPACE-complete), and, moreover, lower than that of well-designed graph patterns with projection on top (Σ_2^p-coomplete).

We proved in Section 7.4 that every well-designed c-query can be transformed into an AUF c-query in polynomial time (actually in logarithmic space). We deduce the following corollary.

COROLLARY 7. *The evaluation problem for well-designed c-queries is* NP-*complete.*

At first this result might seem confusing, since evaluation of well-designed graph patterns is CO-NP-complete. Moreover, this complexity increases to Σ_2^p-coomplete if projection is allowed at the top-most level. But, if instead of SELECT we allow for CONSTRUCT in the top-most level, we obtain a language that is NP-complete. This occurs because when projecting over a set of variables V by means of SELECT, two compatible mappings binding different subsets of V produce two distinct answers. On the other hand, when projecting to a triple pattern t with CONSTRUCT, a mapping that does not bind var(t) will not produce an answer, and two compatible mappings binding var(t) will only produce one answer.

This concludes our study of evaluation complexity for the found fragments. We remark that the results further support that the fragment of AUF c-queries is an appropriate language for querying RDF graphs as it precisely captures the desired semantic notion and at the same time has a low evaluation complexity.

9. CONCLUSIONS AND FUTURE WORK

We presented a thorough study about what is a proper query language for the Semantic Web, in particular considering the importance of querying for optional information. We first showed that the most adopted previous approach (well-designed graph patterns) does not have the desired expressive power. To present a new approach we started by studying the techniques used in first-order logic to capture the semantic notions of monotonicity and closeness under extensions. We then developed a framework for applying these techniques for capturing weakly-monotone graph patterns in SPARQL. This application gave rise to the operator NS, which proved to be a natural way for obtaining optional information. Based on this operator we defined two SPARQL-based languages, which turned out to be better (in terms of expressive power) than previous approaches for querying RDF graphs, in particular better than unions of well-designed graph patterns. Then we focused on the fragment of CONSTRUCT queries. We applied the same techniques from first-order logic that were applied to graph patterns. This application gave us a simple form of queries with the exact desired properties, in terms of both expressive power and computational complexity. The previous results are all relevant to applications and technologies for information published as RDF.

The main conclusion of this work is that the fragment of AUF CONSTRUCT queries is an appropriate query language for RDF. This language produces RDF graphs, a more concise output that allows for relating entities in the answer. Moreover, it has lower evaluation complexity than previous approaches and captures the semantic notion of monotonicity, which derives from the search of a query language that conforms to the open-world-assumption.

Further contributions were shown throughout this dissertation. We presented a simple connection between SPARQL and first-order logic, and a thorough review of the application of Otto's and Lyndon's interpolation theorems to prove preservation theorems. We formalized the fragment of CONSTRUCT queries and discussed the benefits of generating RDF graphs as output (instead of mappings). We proved that AUF CONSTRUCT queries characterize the set of monotone CONSTRUCT queries, and provided an effective algorithm that transforms well-designed CONSTRUCT queries into AUF CONSTRUCT queries. Furthermore, we studied the expressive power and evaluation complexity of several different fragments of SPARQL including the NS operator.

Our study of the presented languages opens new research possibilities, starting by the search for extensions of the defined fragments. For example, allowing for projection on top of simple and ns-patterns preserves weak-monotonicity while increasing the evaluation complexity, which might hint higher expressive power. Also, the defined fragments could be further studied in the context of finite RDF graphs, and lead to finally find a characterization of weakly-monotone graph patterns and monotone CONSTRUCT queries that holds under finite structures. Regarding this last question, it is also important to understand the relation between characterizing monotone CONSTRUCT queries under finite RDF graphs, and monotone first-order formulas under finite models.

Finally, the focus of this dissertation has been mostly theoretical. Therefore, a more practical consideration of the obtained results could lead to a complete new work. For example it is important to understand what are the practical consequences of removing the OPT operator and including the NS operator, or whether the fragment of AUF CONSTRUCT queries could cover the needs of real-world scenarios. Moreover, these new lines of research are open to the development of implementations and optimizations, potentially leading to real applications of the results presented in this work.

References

Abiteboul, S., & Duschka, O. M. (1998). Complexity of answering queries using materialized views. In *Proceedings of the seventeenth acm sigact-sigmod-sigart symposium on principles of database systems* (pp. 254–263).

Abiteboul, S., Hull, R., & Vianu, V. (1995). *Foundations of databases* (Vol. 8). Addison-Wesley Reading.

Ajtai, M., & Gurevich, Y. (1987, October). Monotone versus positive. *J. ACM*, *34*(4), 1004–1015.

Angles, R., & Gutierrez, C. (2008a). The expressive power of SPARQL. In *Iswc* (p. 114-129).

Angles, R., & Gutierrez, C. (2008b). *The expressive power of sparql*. Springer.

Aranda, C. B., Hogan, A., Umbrich, J., & Vandenbussche, P. (2013). SPARQL web-querying infrastructure: Ready for action? In *The semantic web - ISWC 2013 - 12th international semantic web conference, sydney, nsw, australia, october 21-25, 2013, proceedings, part II* (pp. 277–293).

Arenas, M., Conca, S., & Pérez, J. (2012). Counting beyond a yottabyte, or how SPARQL 1.1 property paths will prevent adoption of the standard. In *Proceedings of the 21st international conference on world wide web* (pp. 629–638).

Arenas, M., & Pérez, J. (2011). Querying semantic web data with sparql. In *Proceedings of the thirtieth acm sigmod-sigact-sigart symposium on principles of database systems* (pp. 305–316).

Benedikt, M., ten Cate, B., & Tsamoura, E. (2014). Generating low-cost plans from proofs. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems, pods'14, snowbird, ut, usa, june 22-27, 2014* (pp. 200–211).

Berners-Lee, T., Hendler, J., Lassila, O., et al. (2001). The semantic web. *Scientific american*, *284*(5), 28–37.

Bertossi, L. (2006, June). Consistent query answering in databases. *SIGMOD Rec.*, *35*(2), 68–76.

Bry, F., Furche, T., Marnette, B., Ley, C., Linse, B., & Poppe, O. (2010). Sparqlog: Sparql with rules and quantification. In R. de Virgilio, F. Giunchiglia, & L. Tanca (Eds.), *Semantic web information management* (p. 341-370). Springer Berlin Heidelberg.

Buil-Aranda, C., Arenas, M., & Corcho, O. (2011). Semantics and optimization of the SPARQL 1.1 federation extension. In *The semanic web: Research and applications* (pp. 1–15). Springer.

Buss, S. R., & Hay, L. (1991). On truth-table reducibility to sat. *Information and Computation*, *91*(1), 86 - 102.

Chekol, M. W., Euzenat, J., Genevès, P., & Layaïda, N. (2012a). SPARQL query containment under $\mathcal{SHI}$ axioms. In *Aaai*.

Chekol, M. W., Euzenat, J., Genevès, P., & Layaïda, N. (2012b). SPARQL query containment under RDFS entailment regime. In *Ijcar* (p. 134-148).

Craig, W. (1957). Three uses of the herbrand-gentzen theorem in relating model theory and proof theory. *The Journal of Symbolic Logic*, *22*(3), pp. 269-285.

Durst, M., & Suignard, M. (2005). *Rfc 3987, internationalized resource identifiers (iris).* http://www.ietf.org/rfc/rfc3987.txt.

Enderton, H., & Enderton, H. B. (2001). *A mathematical introduction to logic*. Academic press.

Erling, O., & Mikhailov, I. (2009). RDF support in the virtuoso DBMS. In *Networked knowledge-networked media* (pp. 7–24). Springer.

Fagin, R. (1996). Combining fuzzy information from multiple systems (extended abstract). In *Proceedings of the fifteenth acm sigact-sigmod-sigart symposium on principles of database systems* (pp. 216–226).

Feferman, S. (2008). Harmonious logic: Craig's interpolation theorem and its descendants. *Synthese*, *164*(3), 341–357.

Furche, T., Linse, B., Bry, F., Plexousakis, D., & Gottlob, G. (2006). Rdf querying: Language constructs and evaluation methods compared. In *Reasoning web* (pp. 1–52). Springer.

Galindo-Legaria, C. A. (1994, May). Outerjoins as disjunctions. *SIGMOD Rec.*, *23*(2), 348–358.

Geerts, F., Karvounarakis, G., Christophides, V., & Fundulaki, I. (2013). Algebraic structures for capturing the provenance of SPARQL queries. In *Icdt* (p. 153-164).

Gurevich, Y., & Shelah, S. (1986). Fixed-point extensions of first-order logic. *Annals of Pure and Applied Logic*, *32*, 265 - 280.

Halpin, H., & Cheney, J. (2014). Dynamic provenance for SPARQL updates. In *Iswc*.

Harris, S., Lamb, N., & Shadbolt, N. (2009). 4store: The design and implementation of a clustered rdf store. In *5th international workshop on scalable semantic web knowledge base systems (ssws2009)* (pp. 94–109).

Hemachandra, L. A. (1989). The strong exponential hierarchy collapses. *Journal of Computer and System Sciences*, *39*(3), 299 - 322.

Hernich, A., Libkin, L., & Schweikardt, N. (2011, June). Closed world data exchange. *ACM Trans. Database Syst.*, *36*(2), 14:1–14:40.

Hodges, W. (1997). *A shorter model theory*. New York, NY, USA: Cambridge University Press.

Hogan, A., & Gutierrez, C. (2014). Paths towards the sustainable consumption of semantic data on the web. In *Proceedings of the 8th alberto mendelzon workshop on foundations of data management, cartagena de indias, colombia, june 4-6, 2014.*

Immerman, N. (1982). Relational queries computable in polynomial time. In *Proceedings of the fourteenth annual acm symposium on theory of computing* (pp. 147–152).

Kostylev, E. V., Reutter, J. L., & Ugarte, M. (2015). CONSTRUCT queries in SPARQL. In *18th international conference on database theory, ICDT 2015, march 23-27, 2015, brussels, belgium* (pp. 212–229).

Lenzerini, M. (2002). Data integration: A theoretical perspective. In *Proceedings of the twenty-first acm sigmod-sigact-sigart symposium on principles of database systems* (pp. 233–246).

Letelier, A., Pérez, J., Pichler, R., & Skritek, S. (2012). Static analysis and optimization of semantic web queries. In *Proceedings of the 31st symposium on principles of database systems* (pp. 89–100). New York, NY, USA: ACM.

Letelier, A., Pérez, J., Pichler, R., & Skritek, S. (2013). Static analysis and optimization of semantic web queries. *ACM Trans. Database Syst.*, *38*(4), 25.

Libkin, L. (2006). Data exchange and incomplete information. In *Proceedings of the twenty-fifth acm sigmod-sigact-sigart symposium on principles of database systems* (pp. 60–69). New York, NY, USA: ACM.

Libkin, L. (2013). *Elements of finite model theory*. Springer Science & Business Media.

Libkin, L., Reutter, J., & Vrgoč, D. (2013). TriAL for RDF: adapting graph query languages for RDF data. In *Proceedings of the 32nd symposium on principles of database systems* (pp. 201–212).

Libkin, L., & Sirangelo, C. (2011). Data exchange and schema mappings in open and closed worlds. *Journal of Computer and System Sciences*, *77*(3), 542 - 571. (Database Theory)

Losemann, K., & Martens, W. (2012). The complexity of evaluating path expressions in SPARQL. In *Proceedings of the 31st symposium on principles of database systems* (pp. 101–112).

Lyndon, R. C. (1959). An interpolation theorem in the predicate calculus. *Pacific Journal of Mathematics*, *9*(1), 129–142.

Mallea, A., Arenas, M., Hogan, A., & Polleres, A. (2011). On blank nodes. In *International semantic web conference* (p. 421-437).

Manola, F., & Miller, E. (2004, 10 February). *RDF Primer.* W3C Recommendation. (Available at `http://www.w3.org/TR/2004/REC-rdf-primer-20040210/`)

Nash, A., Segoufin, L., & Vianu, V. (2010, July). Views and queries: Determinacy and rewriting. *ACM Trans. Database Syst.*, *35*(3), 21:1–21:41.

Oberschelp, A. (1968, 06). On the craig-lyndon interpolation theorem. *J. Symbolic Logic*, *33*(2), 271–274.

Otto, M. (2000). An interpolation theorem. *Bulletin of Symbolic Logic*, *6*(4), 447–462.

Papadimitriou, C. H., & Yannakakis, M. (1982). The complexity of facets (and some facets of complexity). In *Proceedings of the fourteenth annual acm symposium on theory of computing* (pp. 255–260). New York, NY, USA: ACM.

Pérez, J., Arenas, M., & Gutierrez, C. (2006a). Semantics and complexity of sparql. In *Iswc* (p. 30-43).

Pérez, J., Arenas, M., & Gutierrez, C. (2006b). Semantics and complexity of sparql. *CoRR*, *abs/cs/0605124*.

Pérez, J., Arenas, M., & Gutierrez, C. (2009). Semantics and complexity of SPARQL. *ACM Trans. Database Syst.*, *34*(3).

Picalausa, F., & Vansummeren, S. (2011). What are real SPARQL queries like? In *Swim.*

Pichler, R., & Skritek, S. (2014). Containment and equivalence of well-designed SPARQL. In *Proceedings of the 33rd acm sigmod-sigact-sigart symposium on principles of database systems* (pp. 39–50).

Polleres, A., Scharffe, F., & Schindlauer, R. (2007). SPARQL++ for mapping between RDF vocabularies. In *On the move to meaningful internet systems 2007: Vilamoura, portugal, november 25-30, 2007, proceedings, part I* (pp. 878–896).

Polleres, A., & Wallner, J. P. (2013). On the relation between SPARQL1.1 and answer set programming. *Journal of Applied Non-Classical Logics*, *23*(1-2), 159-212.

Prud'hommeaux, E., & Seaborne, A. (2008). *SPARQL query language for RDF, W3C recommendation.* **http://www.w3.org/tr/rdf-sparql-query**.

Prud'hommeaux, E., Seaborne, A., et al. (2006). SPARQL query language for RDF.

Ramanathan, V. G. (2013). Light at the end of the tunnel.

Riege, T., & Rothe, J. (2006, may). Completeness in the boolean hierarchy: Exact-four-colorability, minimal graph uncolorability, and exact domatic number problems - a survey., *12*(5), 551–578.

Rossman, B. (2008, August). Homomorphism preservation theorems. *J. ACM*, *55*(3), 15:1–15:53.

Schmidt, M., Meier, M., & Lausen, G. (2010). Foundations of sparql query optimization. In *Proceedings of the 13th international conference on database theory* (pp. 4–33). New York, NY, USA: ACM.

Seaborne, A. (2010). ARQ-A SPARQL processor for Jena. *Obtained through the Internet: http://jena. sourceforge. net/ARQ/*.

Spakowski, H. (2005). *Completeness for parallel access to np and counting class separations* (Unpublished doctoral dissertation).

Tait, W. W. (1959, 3). A counterexample to a conjecture of scott and suppes. *Journal of Symbolic Logic*, *24*, 15–16.

Vardi, M. Y. (1982). The complexity of relational query languages. In *Proceedings of the fourteenth annual acm symposium on theory of computing* (pp. 137–146).

W3C SPARQL Working Group. (2013, 21 March). *SPARQL 1.1 Query language.* W3C Recommendation. (Available at `http://www.w3.org/TR/sparql11-query/`)

Wagner, K. W. (1987, March). More complicated questions about maxima and minima, and some closures of np. *Theor. Comput. Sci.*, *51*(1-2), 53–80.

Wechsung, G. (1985). On the boolean closure of NP. In *Fundamentals of computation theory* (pp. 485–493).