The Challenge of Optional Matching in SPARQL [⋆]

Shqiponja Ahmetaj, Wolfgang Fischl, Markus Kröll, Reinhard Pichler, Mantas Šimkus, and Sebastian Skritek

Database and Artificial Intelligence Group, Faculty of Informatics, TU Wien
{ahmetaj, wfischl, kroell, pichler, simkus, skritek}@dbai.tuwien.ac.at

Abstract. Conjunctive queries are arguably the most widely used querying mechanism in practice and the most intensively studied one in database theory. Answering a conjunctive query (CQ) comes down to matching all atoms of the CQ simultaneously into the database. As a consequence, a CQ fails to provide any answer if the pattern described by the query does not exactly match the data. CQs might thus be too restrictive as a querying mechanism for data on the web, which is considered as inherently incomplete. The semantic web query language SPARQL therefore contains the OPTIONAL operator as a crucial feature. It allows the user to formulate queries which try to match parts of the query over the data if available, but do not destroy answers of the remaining query otherwise. In this article, we have a closer look at this optional matching feature of SPARQL. More specifically, we will survey several results which have recently been obtained for an interesting fragment of SPARQL – the so-called well-designed SPARQL graph patterns.

1 Introduction

Conjunctive queries (or, equivalently, SELECT-FROM-WHERE queries in SQL) are arguably the most widely used querying mechanism in practice and the most intensively studied one in database theory. Answering a conjunctive query (CQ) comes down to matching all atoms of the CQ simultaneously into the database. As a consequence, a CQ fails to provide any answer if the pattern described by the query does not exactly match the data. CQs might thus be too restrictive as a querying mechanism for data on the web, which is considered as inherently incomplete. The semantic web query language SPARQL therefore contains the OPTIONAL operator (abbreviated as OPT henceforth) as a crucial feature. It allows the user to formulate queries which try to match parts of the query over the data if available, but do not destroy answers of the remaining query otherwise. It thus corresponds to the left outer join in the relational algebra. The following example from [24] presents a simple SPARQL query using this feature.

[⋆] This is a pre-print of an article published in Proceedings of Foundations of Information and Knowledge Systems - 9th International Symposium, FoIKS 2016, Linz, Austria, March 7-11, 2016. The final authenticated version is available online at: https://doi.org/10.1007/978-3-319-30024-5_10

Example 1. Consider the following SPARQL query Q which is posed over a database that stores information about movies:[1]

$$Q = \Big(\big((?x, \texttt{directed\_by}, ?y) \,\mathsf{AND}\, (?x, \texttt{released}, \text{“before\_1980”}) \big) \\ \mathsf{OPT}\, (?x, \texttt{oscars\_won}, ?z) \Big) \,\mathsf{OPT}\, (?y, \texttt{first\_movie}, ?z').$$

This query retrieves all pairs (m, d) such that movie m is directed by d and released before 1980. This is specified by the pattern $(?x, \texttt{directed\_by}, ?y) \,\mathsf{AND}\, (?x, \texttt{released}, \text{“before\_1980”})$. Furthermore, whenever possible, this query also retrieves (one or both of) the following pieces of data: the number n of Academy Awards won by movie m and the first movie m' directed by d. In other words, in addition to (m, d) we also retrieve n and/or m' if the information is available in the database. This is specified by the triples $(?x, \texttt{oscars\_won}, ?x)$ and $(?y, \texttt{first\_movie}, ?z')$ following the respective OPT operators. ◇

Apart from AND and OPT used in Example 1, SPARQL also provides the operators UNION and FILTER. SPARQL 1.1 [18] introduces many further operators, which we ignore for the time being. *Projection* is realized by wrapping a SPARQL graph pattern into a SELECT statement where we may explicitly specify the variables of interest. For instance, in Example 1, we could wrap the query Q into a statement of the form $\mathsf{SELECT}\ ?x,\ ?y\ \mathsf{WHERE}\ \{Q\}$ to project out the information on directors and their first movie.

As far as the expressive power of SPARQL is concerned, it was shown in [32, 3] that SPARQL is relational complete. Not surprisingly, the SPARQL query evaluation problem (i.e., given an RDF graph G, a SPARQL query Q, and a set μ of variable bindings, check if μ is a solution) is PSPACE-complete (combined complexity) [29, 35]. The OPT operator was identified as one of the main sources of complexity. Indeed, it was shown in [35] that the PSPACE-completeness of SPARQL query evaluation holds even if we restrict SPARQL to the AND and OPT operator. The reason for this high complexity is the unrestricted use of variables inside and outside an OPT expression. Therefore, in [29], the class of *well-designed* SPARQL graph patterns was introduced. The restriction imposed there is that if a variable occurs on the right-hand side of an OPT expression and anywhere else in the SPARQL graph pattern, then it must also occur on the left-hand side of the OPT expression. It was shown that the complexity of the evaluation problem for the well-designed fragment drops to coNP-completeness [29].

In [29], many further interesting properties of well-designed SPARQL graph patterns were shown. At this point, we mention only one, namely the efficient transformation into so-called OPT *normal form*: a SPARQL graph pattern using AND and OPT operators is in this normal form, if the OPT operator does not occur in the scope of an AND operator. It was shown in [29] that this can always be achieved efficiently by exploiting the equivalence $(P_1\ \mathsf{AND}\ (P_2\ \mathsf{OPT}\ P_3)) \equiv$

[1] We use here the algebraic-style notation from [29] rather than the official SPARQL syntax of [33]. In particular, we explicitly use an AND operator (rather than comma-separated lists) to denote conjunctions.

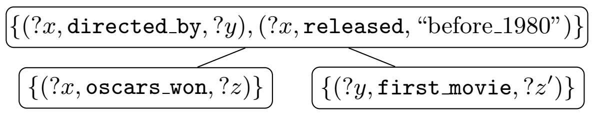

Fig. 1. The wdPT $\mathcal{T}$ representing the query Q from Example 1.

$((P_1 \ \mathsf{AND}\ P_2)\ \mathsf{OPT}\ P_3)$, which holds for well-designed SPARQL graph patterns. Moreover, such graph patterns allow for a natural tree representation, formalized by so-called *well-designed pattern trees* (wdPTs, for short) in [25].

Intuitively, the nodes in a wdPT correspond to CQs while the tree structure represents the optional extensions. For instance, the wdPT corresponding to the query in Example 1 is displayed in Figure 1. As with SPARQL graph patterns, we can add projection to wdPTs by indicating the distinguished variables to which the result shall be projected. Well-designed pattern trees then yield a natural extension of conjunctive queries. Indeed, a CQ corresponds to a wdPT consisting of the root node only. It turns out that the extension of CQs to wdPTs can have a significant effect on various computational tasks. For instance, while query evaluation and query containment are both NP-complete for CQs, these tasks become Σ_2^P-complete [25] or even undecidable [30], respectively, for wdPTs with projection. Actually, it is even questionable if the definition of containment via set inclusion is appropriate in case of optional matching. Also the semantics definition of answering CQs in the presence of ontologies requires rethinking for wdPTs [2].

In this article, we survey these and several further results which have recently been obtained for well-designed SPARQL graph patterns or, equivalently, for well-designed pattern trees. We shall thus mainly concentrate on algorithms and complexity results obtained for the most fundamental computational problems in this area, namely *query evaluation* (see Section 3) and basic *static query analysis* tasks such as testing containment and equivalence of two queries (see Section 4). Finally, we shall also recall results on the evaluation of wdPTs in the presence of ontologies from the DL-Lite family [10] and briefly discuss some unintuitive behavior of SPARQL entailment regimes according to the the recently released W3C recommendation [14] (see Section 5).

2 RDF, SPARQL, and Pattern Trees

RDF. The data model designed for the *Semantic Web* is the Resource Description Framework (RDF) [13]. We focus here on ground RDF graphs and assume them to be composed of URIs only. Formally, let $\mathbf{U}$ be an infinite set of URIs. An *RDF triple* (s, p, o) is a tuple in $\mathbf{U} \times \mathbf{U} \times \mathbf{U}$, whose components are referred to as subject, predicate, and object, respectively. An *RDF graph* is a finite set of RDF triples. Note that a set of triples (s, p, o) can be seen as an edge-labeled graph, where s and o denote vertices and p denotes an edge label. The active domain $\mathsf{dom}(G) \subseteq \mathbf{U}$ of an RDF graph G is the set of URIs actually appearing in G.

SPARQL syntax. SPARQL [33], which was later extended to SPARQL 1.1 [18], is the standard query language for RDF data. Following the presentation in [29], we next recall the formalization of its graph pattern matching facility, which forms the core of the language. Let $\mathbf{V}$ be an infinite set of variables with $\mathbf{U} \cap \mathbf{V} = \emptyset$. We write variables in $\mathbf{V}$ with a leading question mark, as in $?x$. A *SPARQL triple pattern* is a tuple in $(\mathbf{U} \cup \mathbf{V}) \times (\mathbf{U} \cup \mathbf{V}) \times (\mathbf{U} \cup \mathbf{V})$. More complex patterns studied in this article are constructed using the operators AND, OPT, and UNION. We omit here further operators specified by [33] and [18], including the FILTER operator. Formally, *SPARQL graph patterns* (or simply *graph patterns*, for short) are thus recursively defined as follows. (1) a triple pattern is a graph pattern, and (2) if P_1 and P_2 are graph patterns, then $(P_1 \circ P_2)$ is a graph pattern for $\circ \in \{\mathsf{AND}, \mathsf{OPT}, \mathsf{UNION}\}$. Let P be a graph pattern or a set of graph patterns; then we write $\mathsf{vars}(P)$ to denote the set of variables occurring in P.

SPARQL semantics. For defining the semantics of SPARQL graph patterns, we again follow closely the definitions proposed in [29]. A *mapping* is a function $\mu \colon A \to \mathbf{U}$ for some $A \subset \mathbf{V}$. For a triple pattern t with $\mathsf{vars}(t) \subseteq \mathsf{dom}(\mu)$, we write $\mu(t)$ to denote the triple obtained by replacing the variables in t according to μ. Two mappings μ_1 and μ_2 are called compatible (written $\mu_1 \sim \mu_2$) if $\mu_1(?x) = \mu_2(?x)$ for all $?x \in \mathsf{dom}(\mu_1) \cap \mathsf{dom}(\mu_2)$. A mapping μ_1 is subsumed by μ_2 (written $\mu_1 \sqsubseteq \mu_2$) if $\mu_1 \sim \mu_2$ and $\mathsf{dom}(\mu_1) \subseteq \mathsf{dom}(\mu_2)$. In this case, we also say that μ_2 is an extension of μ_1. Subsumption is naturally extended to sets of mappings, e.g., $\mu \sqsubseteq M$ for a set M of mappings, if $\mu \sqsubseteq \mu'$ for some $\mu' \in M$.

We formalize the evaluation of graph patterns over an RDF graph G as a function $[\![\cdot]\!]_G$ that, given a graph pattern P, returns a set of mappings (i.e., the "solutions" or "answers" of P over G). It is defined recursively as follows [29]:

1. $[\![t]\!]_G = \{\mu \mid \mathsf{dom}(\mu) = \mathsf{vars}(t) \text{ and } \mu(t) \in G\}$ for a triple pattern t.
2. $[\![P_1 \ \mathsf{AND}\ P_2]\!]_G = \{\mu_1 \cup \mu_2 \mid \mu_1 \in [\![P_1]\!]_G, \mu_2 \in [\![P_2]\!]_G, \text{ and } \mu_1 \sim \mu_2\}$.
3. $[\![P_1 \ \mathsf{OPT}\ P_2]\!]_G = [\![P_1 \ \mathsf{AND}\ P_2]\!]_G \cup \{\mu_1 \in [\![P_1]\!]_G \mid \forall \mu_2 \in [\![P_2]\!]_G \colon \mu_1 \not\sim \mu_2\}$.
4. $[\![P_1 \ \mathsf{UNION}\ P_2]\!]_G = [\![P_1]\!]_G \cup [\![P_2]\!]_G$.

Note that, as in [29], we assume set semantics, while the W3C Recommendation specifies bag-semantics [33].

Well-designed SPARQL. In [29], the authors identify several classes of graph patterns. One of these classes, which is at the heart of this survey, is formed by the so-called *well-designed* SPARQL graph patterns. A graph pattern P built only from AND and OPT is *well-designed* if there does not exist a subpattern $P' = (P_1 \ \mathsf{OPT}\ P_2)$ of P and a variable $?x$, such that $?x$ occurs in P_2 and in P outside P', but not in P_1. A graph pattern $P = P_1 \ \mathsf{UNION} \ldots \mathsf{UNION}\ P_n$ is well-designed if each subpattern P_i is UNION-free and well-designed. Thus, as in [29], when including the UNION operator, we only allow it to appear outside the scope of other operators.

Well-designed pattern trees. We have already mentioned above the OPT*normal form* [29], which forbids occurrences of the OPT operator in the scope of an AND operator. Well-designed graph patterns in OPT normal form allow for a natural tree representation, formalized by so-called *well-designed pattern trees (wdPTs)* in [25]. A wdPT $\mathcal{T}$ is a pair $(T, \mathcal{P})$ where $T = (V, E, r)$ is a rooted, unordered, tree and $\mathcal{P} = (P_n)_{n \in V}$ is a labeling of the nodes in V, s.t. P_n is a non-empty set of triple patterns (or, equivalently, a conjunction of triple patterns) for every $n \in V$. The *well-designedness* condition requires that, for every variable $?x$ occurring in $\mathcal{T}$, the nodes $\{n \in V \mid ?x \text{ occurs in } P_n\}$ must induce a connected subgraph of T. For instance, the graph pattern Q in Example 1 is in OPT normal form. Its corresponding pattern tree $\mathcal{T}$ is shown in Figure 1.

Graph patterns in OPT normal form consist of conjunctive parts (represented by the nodes of the pattern tree) that are located in a structure of nested OPT operators (modeled by the tree-structure). Note that the order of child nodes in such a tree does not matter. This is due to equivalence $\big((P_1 \ \mathsf{OPT}\ P_2)\ \mathsf{OPT}\ P_3\big) \equiv \big((P_1 \ \mathsf{OPT}\ P_3)\ \mathsf{OPT}\ P_2\big)$, which holds for well-designed graph patterns [29]. This is why wdPTs are defined as *unordered* trees.

Components of a pattern tree. Let $\mathcal{T} = ((V, E, r), \mathcal{P})$ be a wdPT. We call a wdPT $\mathcal{T}' = ((V', E', r'), (P_n)_{n \in V'})$ a *subtree of* $\mathcal{T}$ if (V', E', r') is a subtree of T. $\mathcal{T}'$ is a *subtree of* $\mathcal{T}$ *containing the root* if $r' = r$. Throughout this article, unless explicitly specified otherwise, we always consider subtrees containing the root, and will thus refer to them simply as "subtrees", omitting the phrase "containing the root". An *extension* $\hat{\mathcal{T}}'$ of a subtree $\mathcal{T}'$ of $\mathcal{T}$ is a subtree $\hat{\mathcal{T}}'$ of $\mathcal{T}$, s.t. $\mathcal{T}'$ is in turn a subtree of $\hat{\mathcal{T}}'$. A subtree or extension is *proper* if some node of the bigger tree is missing in the smaller tree.

Given a wdPT $\mathcal{T} = ((V, E, r), \mathcal{P})$, we write $V(\mathcal{T})$ to denote the set V of vertices. We sometimes refer to the set P_n of triple patterns at vertex $n \in V$ as $\mathit{pat}(n)$ and we denote by $\mathit{pat}(\mathcal{T})$ the set $\bigcup_{n \in V(\mathcal{T})} P_n$ of triple patterns occurring in $\mathcal{T}$. We write $\mathsf{vars}(\mathcal{T})$ (resp. $\mathsf{vars}(n)$) as an abbreviation for $\mathsf{vars}(\mathit{pat}(\mathcal{T}))$ (resp. $\mathsf{vars}(\mathit{pat}(n))$. These notions extend naturally to sets of nodes. For nodes $n, \hat{n} \in V(\mathcal{T})$, s.t. $\hat{n}$ is the parent of n, let $\mathsf{newvars}(n) = \mathsf{vars}(n) \setminus \mathsf{vars}(\hat{n})$. A wdPT $\mathcal{T}$ is said to be in *NR normal form*, if $\mathsf{newvars}(n) \neq \emptyset$ for every $n \in V(\mathcal{T})$ [25]. It was shown in [25], that every wdPT can be transformed efficiently into an equivalent wdPT in NR normal form. We therefore assume w.l.o.g. that all wdPTs dealt with here are in NR normal form.

Semantics of pattern trees. Analogously to graph patterns, the result of evaluating a wdPT $\mathcal{T}$ over some RDF graph G is denoted by $[\![\mathcal{T}]\!]_G$. In [25], the set $[\![\mathcal{T}]\!]_G$ of solutions was defined via a translation to graph patterns. However, for wdPTs in NR normal form, the set of solutions $[\![\mathcal{T}]\!]_G$ has a nice direct characterization in terms of maximal subtrees of $\mathcal{T}$:

Lemma 1 ([25]). *Let $\mathcal{T}$ be a wdPT in NR normal form and G an RDF graph. Then $\mu \in [\![\mathcal{T}]\!]_G$ iff there exists a subtree $\mathcal{T}'$ of $\mathcal{T}$, s.t. (1) $\mathsf{dom}(\mu) = \mathsf{vars}(\mathcal{T}')$, and (2) $\mathcal{T}'$ is the maximal subtree of $\mathcal{T}$, s.t. $\mu \sqsubseteq [\![\mathit{pat}(\mathcal{T}')]\!]_G$.*

It can be easily checked that $\mathcal{T}'$ is uniquely defined by $\mathsf{dom}(\mu)$. We refer to this tree as $\mathcal{T}_\mu$. We illustrate the evaluation of graph patterns or, equivalently, of wdPTs) by revisiting Example 1.

Example 2. Consider the following RDF graph G:

$$\begin{aligned} G = \{ & (\text{“American\_Graffiti”}, \texttt{directed\_by}, \text{“George\_Lucas”}),\\ & (\text{“American\_Graffiti”}, \texttt{released}, \text{“before\_1980”}),\\ & (\text{“Star\_Wars”}, \texttt{directed\_by}, \text{“George\_Lucas”}),\\ & (\text{“Star\_Wars”}, \texttt{released}, \text{“before\_1980”}),\\ & (\text{“Star\_Wars”}, \texttt{oscars\_won}, \text{“6”})\}. \end{aligned}$$

The evaluation of the query Q from Example 1 (or, equivalently of the wdPT $\mathcal{T}$ in Figure 1) over G, yields the partial mappings μ_1 and μ_2 defined on the variables $?x, ?y, ?t$, and $?z'$ such that: (1) $\mathsf{dom}(\mu_1) = \{?x, ?y\}$ with $\mu_1 = \{?x \leftarrow$ “American_Graffiti”$, ?y \leftarrow$ “George_Lucas”$\}$ and (2) $\mathsf{dom}(\mu_2) = \{?x, ?y, ?z\}$ with $\mu_2 = \{?x \leftarrow$ “Star_Wars”$, ?y \leftarrow$ “George_Lucas”$, ?z \leftarrow$ “6”$\}$. ◇

Projection. Recall that projection is not considered as part of a graph pattern [33]; instead, it is realized by the SELECT result modifier on top of a graph pattern. For a mapping μ and a set $\mathcal{X}$ of variables, let $\mu_{|\mathcal{X}}$ denote the projection of μ to the variables in $\mathcal{X}$, that is, the mapping μ' defined as $\mathsf{dom}(\mu') := \mathcal{X} \cap \mathsf{dom}(\mu)$ and $\mu'(?x) := \mu(?x)$ for all $?x \in \mathsf{dom}(\mu')$.

It is convenient to denote a graph pattern P or a wdPT $\mathcal{T}$ with projection to $\mathcal{X}$ as $(P, \mathcal{X})$ and $(\mathcal{T}, \mathcal{X})$, respectively. The evaluation of such a graph pattern or wdPT over an RDF graph G is then defined as $[\![(P, \mathcal{X})]\!]_G = \{\mu_{|\mathcal{X}} \mid \mu \in [\![P]\!]_G\}$ and $[\![(\mathcal{T}, \mathcal{X})]\!]_G = \{\mu_{|\mathcal{X}} \mid \mu \in [\![\mathcal{T}]\!]_G\}$, respectively. We refer to the pair $(\mathcal{T}, \mathcal{X})$ as a *wdPT with projection* or simply a *wdPT*, for short. Moreover, we refer to $\mathsf{vars}(\mathcal{T}) \cap \mathcal{X}$ as the free variables ($\mathsf{fvars}(\mathcal{T})$) and to $\mathsf{vars}(\mathcal{T}) \setminus \mathsf{fvars}(\mathcal{T})$ as the existential variables in $\mathcal{T}$ ($\mathsf{evars}(\mathcal{T})$). Analogously, we write $\mathsf{fvars}(n)$ and $\mathsf{evars}(n)$, respectively, for nodes $n \in V(\mathcal{T})$. Moreover, for $n \in V(\mathcal{T})$, let $\mathsf{newfvars}(n) = \mathsf{newvars}(n) \cap \mathsf{fvars}(n)$. W.l.o.g., we assume that existential variables in wdPTs with projection are always renamed apart, i.e., $\mathsf{evars}(\mathcal{T}_1) \cap \mathsf{evars}(\mathcal{T}_2) = \emptyset$ for any two distinct wdPTs $\mathcal{T}_1$ and $\mathcal{T}_2$.

A wdPT $(\mathcal{T}, \mathcal{X})$ is in NR normal form if $\mathcal{T}$ is. For wdPTs with projection, a similar characterization of solutions as Lemma 1 exists.

Lemma 2 ([25]). *Let $(\mathcal{T}, \mathcal{X})$ be a wdPT with projection in NR normal form, G an RDF graph and μ a mapping with $\mathsf{dom}(\mu) \subseteq \mathcal{X}$. Then $\mu \in [\![(\mathcal{T}, \mathcal{X})]\!]_G$ iff there exists a subtree $\mathcal{T}'$ of $\mathcal{T}$, s.t. (1) $\mathsf{dom}(\mu) = \mathsf{fvars}(\mathcal{T}')$, and (2) there exists a mapping $\lambda \colon \mathsf{evars}(\mathcal{T}') \to \mathsf{dom}(G)$, s.t. $\mu \cup \lambda \in [\![\mathcal{T}]\!]_G$.*

SPARQL allows the use of blank nodes in graph patterns (see [18] for details), which we do not consider here. This is however no restriction, since every well-designed graph pattern with blank nodes is equivalent to a well-designed graph pattern with projection but without blank nodes.

Union. Recall that we allow the UNION operator only to be applied “top-level”, i.e., well-designed SPARQL graph patterns involving the UNION operator are of the form $P = P_1$ UNION $\ldots$ UNION P_k, such that each P_i is a UNION-free well-designed graph pattern. Analogously, we consider a set $\{\mathcal{T}_1, \ldots, \mathcal{T}_k\}$ of wdPTs (i.e., a *well-designed pattern forest (wdPF)*) with the intended meaning that it stands for the union of the wdPTs. All notions introduced for wdPTs extend naturally to wdPFs, e.g., a subtree $\mathcal{T}'$ of a wdPF $\mathcal{F}$ is a subtree for some wdPT $\mathcal{T} \in \mathcal{F}$. We define the set of solutions of a wdPF $\mathcal{F}$ without projection and of a wdPF $(\mathcal{F}', \mathcal{X})$ with projection over an RDF graph G as $[\![\mathcal{F}]\!]_G := \bigcup_{\mathcal{T} \in \mathcal{F}} [\![\mathcal{T}]\!]_G$ and $[\![(\mathcal{F}', \mathcal{X})]\!]_G := \bigcup_{(\mathcal{T}, \mathcal{X}) \in \mathcal{F}'} [\![(\mathcal{T}, \mathcal{X})]\!]_G$, respectively.

3 Query Evaluation

In this section we have a closer look at the evaluation of well-designed SPARQL graph patterns or, equivalently, of wdPTs. To this end, we first revisit the semantics definition of SPARQL graph patterns or wdPTs from Section 2. Recall that the semantics $[\![\cdot]\!]_G$ is inductively defined over the structure of SPARQL graph patterns. In terms of wdPTs, a direct implementation of this semantics definition corresponds to a bottom-up traversal of the tree. Clearly, it may thus happen that one computes big intermediate results for some subtree (not containing the root) of the wdPT, which ultimately have to be deleted since these intermediate results cannot be extended to mappings up to the root of the wdPT. For instance, suppose that in Example 2, the graph G is augmented by triples $(\mathrm{d}_1, \texttt{first\_movie}, \mathrm{m}_1)$, $(\mathrm{d}_2, \texttt{first\_movie}, \mathrm{m}_2)$, etc. Then the evaluation of the pattern $\{(?y, \texttt{first\_movie}, ?z')\}$ at the right leaf node of $\mathcal{T}$ yields the mappings $\nu_1 = \{?y \leftarrow \mathrm{d}_1, ?z' \leftarrow \mathrm{m}_1\}$, $\nu_2 = \{?y \leftarrow \mathrm{d}_2, ?z' \leftarrow \mathrm{m}_2\}$, etc. Obviously, none of these mappings can be extended further up to the root node.

In [28] the authors therefore proposed a top-down evaluation method for well-designed SPARQL graph patterns, which avoids the computation of "useless" intermediate results, i.e.: every partial mapping produced by this evaluation method is indeed a solution or can be extended to a solution. Below we illustrate this top-down evaluation for wdPTs [25].

Top-down evaluation. Lemma 1 essentially states that the solutions of a wdPT over some graph G are exactly those mappings which map all triples in some subtree $\mathcal{T}'$ of $\mathcal{T}$ into G, and which cannot be extended to some bigger subtree $\mathcal{T}''$ of $\mathcal{T}$. This characterization inspires the following procedural semantics that is obtained by evaluating the pattern tree via a top-down traversal. Given a label P_n of node n in $\mathcal{T}$ and a graph G, we denote by $[\![P_n]\!]_G$ the set of mappings μ that send all triples in P_n into G, i.e., $[\![P_n]\!]_G = \{\mu \mid \mu(t) \in G \text{ for all } t \in P_n\}$.

Definition 1. *Consider an RDF graph G, a wdPT $\mathcal{T} = ((V, E, r), \mathcal{P})$ with $\mathcal{P} = (P_n)_{n \in V}$, and a set M of mappings. For $n \in V$, we define the evaluation of $\mathcal{T}_n$ (the complete subtree of $\mathcal{T}$ rooted at n) given M over G, denoted by* $\mathrm{ext}(M, n, G)$ *as follows. If n is a leaf, then*

$$\mathrm{ext}(M, n, G) = M \bowtie [\![P_n]\!]_G,$$

and, otherwise, if $n_1, \ldots, n_k$ are the child nodes of n, then

$$\mathrm{ext}(M, n, G) = M_1 \bowtie M_2 \bowtie \cdots \bowtie M_k,$$

where $M_i = (M \bowtie [\![P_n]\!]_G) \rtimes \mathrm{ext}(M \bowtie [\![P_n]\!]_G, n_i, G)$. *We define the* top-down evaluation *of $\mathcal{T}$ over G, denoted by $[\![\mathcal{T}]\!]_G^{td}$, as*

$$[\![\mathcal{T}]\!]_G^{td} = \mathrm{ext}(\{\mu_\emptyset\}, r, G),$$

where $\mu_\emptyset$ is the mapping with the empty domain.

The above definition can be also seen in a more procedural way: Given some wdPT with root r and some RDF graph G, first get the set M of all mappings that map P_r into G. For each mapping $\mu \in M$ property (1) of Lemma 1 is satisfied. Now in order to test property (2), it suffices to check for each such mapping μ if it can be extended to some child n of r, i.e. to some mapping $\mu' \colon \mathsf{vars}(P_r) \cup \mathsf{vars}(P_n) \to \mathsf{dom}(G)$ with $\mu'(P_n) \subseteq G$. If this is possible, replace μ by μ'. Note that μ' again satisfies property (1) of Lemma 1. Hence one way to think of this evaluation method is to maintain a set of partial solutions together with a subtree $\mathcal{T}'$ of the input wdPT rooted at r for each of them. In order to determine whether the mapping can be extended, it suffices to check if it can be extended to a child node of the leaf nodes of $\mathcal{T}'$.

The following theorem states that the top-down evaluation defined above coincides with the semantics of pattern trees recalled in Section 2.

Theorem 1 ([25]). *Let $\mathcal{T}$ be a wdPT and G an RDF graph. Then $[\![\mathcal{T}]\!]_G = [\![\mathcal{T}]\!]_G^{td}$.*

Complexity of evaluation without projection. We now look at the complexity of the EVALUATION problem of SPARQL graph patterns or, equivalently, of wdPTs. We thus study the following decision problem: Given a wdPT $\mathcal{T}$, an RDF graph G, and a mapping μ, check if μ is a solution. For wdPTs without projection, it was shown in in [29] that this problem is coNP-complete. For our representation of SPARQL graph patterns as wdPTs in NR normal form, a coNP test can work as follows. Let $\mathcal{T} = ((V, E, r), \mathcal{P})$ be a wdPT. By using the characterization of the evaluation of wdPTs provided in Lemma 1, in order to check whether μ is a solution of $\mathcal{T}$ over G, the coNP-algorithm can first find a subtree $\mathcal{T}'$ of $\mathcal{T}$ rooted at r s.t. $\mathsf{dom}(\mu) = \mathsf{vars}(\mathcal{T}')$. Notice that if this subtree exists, then it is unique (since $\mathcal{T}$ is in NR normal form), and thus, this step can be done in polynomial time. Then the algorithm checks that $\mathcal{T}'$ is a maximal subtree such that $\mu \sqsubseteq [\![pat(\mathcal{T}')]\!]_G$. The latter test requires coNP-power since we have to check that μ cannot be extended to any of the sets of triple patterns at nodes in $\mathcal{T}$, which are "below" the leaf nodes of $\mathcal{T}'$. However, it is sufficient to check this for every child node of $\mathcal{T}'$ (i.e., for every child in $\mathcal{T}$ of a leaf node of $\mathcal{T}'$) individually: if μ can be extended to any child node, this immediately proves that it is not maximal. Note that this simple coNP-algorithm heavily relies on the NR normal form; the coNP-algorithm provided in [29] is considerably more involved.

Now consider the relationship with CQs. Clearly, if all sets of triple patterns are from tractable fragments of CQ evaluation, then the problem of checking whether μ is a solution of $\mathcal{T}$ over G also becomes tractable. This follows immediately from the algorithm sketched above: instead of coNP-power to test whether μ cannot be extended to any "child" node of $\mathcal{T}'$, this is now feasible in polynomial time. Note that tractability is required for each set P_n individually, hence for different nodes n and n', the sets P_n and $P_{n'}$ may belong to different tractable fragments.

Complexity of evaluation with projection. For CQs without existentially quantified variables, the decision problem corresponding to EVALUATION is tractable. However, it becomes NP-complete for CQs with existentially quantified variables. The next result shows that a similar behavior can be observed for well-designed SPARQL graph patterns as well. I.e., the complexity increases by one level in the polynomial hierarchy if projection is added.

Theorem 2 ([25]). *The* EVALUATION *problem of wdPTs with projection is Σ_2^P-complete.*

The membership is shown by devising a simple "guess and check" algorithm that tests whether the solution candidate μ satisfies Lemma 2. Given a wdPT $\mathcal{T}$, a mapping μ, a set $\mathcal{X}$ of free variables, and an RDF graph G, the witness that must be guessed by the algorithm consists of

1. the subtree $\mathcal{T}'$ of $\mathcal{T}$ rooted at r and
2. the mapping λ on $\mathsf{evars}(\mathcal{T}')$.

For the "check" part, it remains to test whether $\mathsf{fvars}(\mathcal{T}') = \mathsf{dom}(\mu)$ and whether $\mu \cup \lambda \in [\![\mathcal{T}]\!]_G$. The first test can be obviously done in polynomial time, while the second test is in coNP [29].

Tractable evaluation. A condition that has been shown to help identifying relevant tractable fragments of wdPTs is *local tractability* [25]. This refers to restricting the CQ defined by each node in a wdPT to belong to a tractable class. The classes of CQ patterns which admit an efficient evaluation include classes of bounded *treewidth* [12], *hypertreewidth*, [15] (generalizing acyclic CQs [36]), *fractional* hypertreewidth, [17], etc. We concentrate on the first two. From now on, we denote by $\mathsf{TW}(k)$ (resp., $\mathsf{HW}(k)$), for $k \geq 1$, the class of CQs of treewidth (resp., hypertreewidth) at most k. A wdPT $((V, E, r), (P_n)_{n \in V})$ is *locally in $\mathcal{C}$*, if for each node $n \in V$ the CQ $ANS \leftarrow P_n$ is in $\mathcal{C}$. We write ℓ-$\mathcal{C}$ for the set of all wdPTs that are locally in $\mathcal{C}$. Moreover we denote by EVAL($\mathcal{C}$) the evaluation problem of wdPTs restricted to the class $\mathcal{C}$.

As already mentioned before, local tractability leads to tractability of evaluation for projection-free wdPTs. On the other hand, this result does not hold in the presence of projection, even when $\mathcal{C}$ is of bounded treewidth. Formally, EVAL(ℓ-$\mathsf{TW}(k)$) and EVAL(ℓ-$\mathsf{HW}(k)$) are NP-complete for every $k \geq 1$ [25].

This raises the question of which further restrictions on wdPTs are needed to achieve tractability. In [8], a natural such restriction is introduced, called *bounded interface*. Intuitively, this restricts the number of variables shared between a node in a wdPT and its children. We say that a wdPT has *c-bounded interface*, for $c \geq 1$, if for each node n of the wdPT, the number of variables that appear both in n and its children is at most c. We denote by $\mathsf{BI}(c)$ the set of wdPTs of c-bounded interface. It can be shown that local tractability and bounded interface yield tractability of the EVALUATION problem of wdPTs with projection:

Theorem 3 ([8]). *Let $\mathcal{C}$ be* $\mathsf{TW}(k)$ *or* $\mathsf{HW}(k)$ *and* $c \geq 1$*. Then* $\textsc{Eval}(\ell\text{-}\mathcal{C} \cap \mathsf{BI}(c))$ *is in* $\textsc{Ptime}$*.*

Notice that CQs are a special case of wdPTs consisting of the root node only. Hence, $\mathsf{TW}(k) \subseteq \ell\text{-}\mathsf{TW}(k) \cap \mathsf{BI}(c)$ and $\mathsf{HW}(k) \subseteq \ell\text{-}\mathsf{HW}(k) \cap \mathsf{BI}(c)$ hold for each $c \geq 1$. Therefore, Theorem 3 tells us that $\ell\text{-}\mathsf{TW}(k) \cap \mathsf{BI}(c)$ and $\ell\text{-}\mathsf{HW}(k) \cap \mathsf{BI}(c)$ define relevant extensions of $\mathsf{TW}(k)$ and $\mathsf{HW}(k)$, respectively, that preserve tractability of evaluation.

Partial evaluation of wdPTs. Given the nature of wdPTs, it is also interesting to check whether a mapping μ is a *partial* solution of the wdPT $\mathcal{T}$ over G [29], i.e., whether μ can be extended to some solution μ' of $\mathcal{T}$ over G. This gives rise to the partial evaluation problem $\textsc{Partial-Eval}(\mathcal{C})$ for a class $\mathcal{C}$ of wdPTs defined as follows: Given a graph G and a wdPT $\mathcal{T} \in \mathcal{C}$, as well as a partial mapping $\mu : \mathcal{X} \to \mathbf{U}$, where $\mathcal{X}$ is the set of variables mentioned in $\mathcal{T}$, is there a $\mu' \in [\![\mathcal{T}]\!]_G$ such that μ' extends μ?

Partial evaluation is tractable for the class of projection-free wdPTs [29]. On the other hand, if projection is allowed, then partial evaluation is NP-complete even under local tractability, i.e., even for the classes $\ell\text{-}\mathsf{TW}(k)$ and $\ell\text{-}\mathsf{HW}(k)$, for each $k \geq 1$ [25].

It is easy to modify the proof of Theorem 3 to show that adding bounded interface to local tractability yields efficient partial evaluation; that is, $\textsc{Partial-Eval}(\ell\text{-}\mathsf{TW}(k) \cap \mathsf{BI}(c))$ and $\textsc{Partial-Eval}(\ell\text{-}\mathsf{HW}(k) \cap \mathsf{BI}(c))$ are in $\textsc{Ptime}$. However, partial evaluation is seemingly easier than exact evaluation. Hence, the question naturally arises if tractability of partial evaluation of wdPTs can be ensured by a weaker condition. Indeed, we give a positive answer to this question below. This condition will be referred to as *global tractability.* Intuitively, it states that there is a bound on the treewidth (resp., hypertreewidth) of the CQs defined by the different subtrees of a wdPT $\mathcal{T}$ rooted in r. Formally, let $\mathcal{C}$ be $\mathsf{TW}(k)$ or $\mathsf{HW}(k)$, for $k \geq 1$. A wdPT $\mathcal{T}$ is *globally in* $\mathcal{C}$, if for each subtree T' of T rooted in r it is the case that the CQ $ANS \leftarrow pat(\mathcal{T}')$ is in $\mathcal{C}$. We denote by g-$\mathcal{C}$ the set of all wdPTs that are globally in $\mathcal{C}$.

Theorem 4 ([8]). $\textsc{Partial-Eval}(g\text{-}\mathsf{TW}(k))$ *and* $\textsc{Partial-Eval}(g\text{-}\mathsf{HW}(k))$ *are in* $\textsc{Ptime}$ *for every* $k \geq 1$*.*

It remains to answer the question if global tractability also suffices to ensure tractability of (exact) evaluation for wdPTs. It turns out that this is not the case.

Proposition 1 ([8]). $\textsc{Eval}(g\text{-}\mathsf{TW}(k))$ *and* $\textsc{Eval}(g\text{-}\mathsf{HW}(k))$ *are* NP*-complete for all* $k \geq 1$*.*

Semantics based on maximal mappings. The semantics of projection-free wdPTs is only based on *maximal* mappings, i.e., mappings that are not subsumed by any other mapping in the answer. This is no longer the case in the presence of projection [25]. As we will see in Section 5, for query answering of SPARQL under entailment regimes, it will turn out advantageous to define a semantics for wdPTs that is based on maximal mappings. This semantics is formalized as follows. Assume G is an RDF Graph and $\mathcal{T}$ is a wdPT. The *evaluation of $\mathcal{T}$ over G under maximal mappings*, denoted $[\![\mathcal{T}]\!]^m_G$, corresponds to the restriction of $[\![\mathcal{T}]\!]_G$ to those mappings μ which are not extended by any other mapping $\mu' \in [\![\mathcal{T}]\!]_G$. This naturally leads to the decision problem MAX-EVAL($\mathcal{C}$) defined as follows: Given an RDF graph G and a wdPT $\mathcal{T} \in \mathcal{C}$, as well as a partial mapping $\mu : \mathcal{X} \to \mathbf{U}$, where $\mathcal{X}$ is the set of variables mentioned in $\mathcal{T}$, is $\mu \in [\![\mathcal{T}]\!]^m_G$?

MAX-EVAL($\mathcal{C}$) is clearly intractable for the class $\mathcal{C}$ of all wdPTs. Analogously to PARTIAL-EVAL, local tractability is not sufficient to ensure tractability of MAX-EVAL:

Proposition 2 ([8]). *For every $k \geq 1$ the problems* MAX-EVAL(ℓ-TW(k)) *and* MAX-EVAL(ℓ-HW(k)) *are* NP*-hard.*

To obtain tractability in this case it is however sufficient to impose global tractability, which is exactly the same condition that yields tractability of partial evaluation for wdPTs (as stated in Theorem 4):

Theorem 5 ([8]). MAX-EVAL(g-TW(k)) *and* MAX-EVAL(g-HW(k)) *are in* PTIME *for every $k \geq 1$.*

4 Static Query Analysis

Static query analysis is a fundamental task in query optimization. Two of the most important problems in this context are query containment and query equivalence, which are very well understood for a variety of query languages [1]. For instance, since by Trakhtenbrot's theorem both problems are undecidable for the full relational calculus, they have been studied for several interesting fragments of relational calculus, including CQs and several extensions thereof [11, 34, 21].

Since SPARQL has the same expressive power as the relational calculus and queries from one language can be effectively transformed into equivalent queries of the other language [32, 3], containment and equivalence are undecidable for full SPARQL. Hence, analogously to the relational calculus, both problems have been studied for fragments of SPARQL, with well-designed graph patterns being the core fragment.

We use the notation wd-SPARQL[S] to refer to the different classes of SPARQL queries reviewed in this section, where $S \subseteq \{\cup, \pi\}$. I.e., we consider well-designed SPARQL queries which use the AND and OPT operator and which may be extended by UNION (if $\cup \in S$) and/or projection (if $\pi \in S$). We will consider the problems CONTAINMENT[S_1,S_2] and EQUIVALENCE[S_1,S_2], which take as input queries $Q_1 \in$ wd-SPARQL[S_1], $Q_2 \in$ wd-SPARQL[S_2] and ask if for all RDF graphs G it is the case that $[\![Q_1]\!]_G \subseteq [\![Q_2]\!]_G$ or $[\![Q_1]\!]_G = [\![Q_2]\!]_G$, respectively, holds.

It was argued that in the presence of optional matching, the classical notion of query containment via the subset relation ($\subseteq$) might be too restrictive for certain applications. The reason for this is illustrated by the following example.

Example 3. Consider the two SPARQL queries $Q_1 = (?x, \texttt{directed\_by}, ?y)$ and $Q_2 = (?y, \texttt{directed\_by}, ?y)\,\mathsf{OPT}\,(?x, \texttt{oscars\_won}, ?z)$ which are simplified variants of the query Q from Example 1, and an RDF graph $G = \{(\text{“Star\_Wars”}, \texttt{directed\_by}, \text{“George\_Lucas”}), (\text{“Star\_Wars”}, \texttt{oscars\_won}, \text{“6”})\}$. Then $[\![Q_1]\!]_G = \{\mu\}$ with $\mu = \{?x \leftarrow \text{“Star\_Wars”}, (?y \leftarrow \text{“George\_Lucas”}\}$ and $[\![Q_2]\!]_G = \{\mu'\}$ with $\mu' = \mu \cup \{?z \leftarrow \text{“6”}\}$. Hence $Q_1 \not\subseteq Q_2$. This might be, however, unintuitive or even unintended, since answers to Q_2 always contain at least the same amount of information as those to Q_1. $\diamond$

One way to address this concern is to resort to the subsumption relation mentioned in Section 2. This gives rise to the problem SUBSUMPTION[S_1,S_2] which, given two queries $Q_1 \in$ wd-SPARQL[S_1] and $Q_2 \in$ wd-SPARQL[S_2], asks if $[\![Q_1]\!]_G \sqsubseteq [\![Q_2]\!]_G$ holds for all RDF graphs G. Clearly, for CQs, the notions of containment and subsumption coincide. Subsumption has already been used in the past as a meaningful way of testing containment of queries with incomplete answers over semistructured data [20], and it has been convincingly argued that it is also a suitable notion for comparing the result of SPARQL queries containing the OPT operator [5]. It has also been considered in foundational work on SPARQL to compare the evaluation of two patterns containing OPT operators [29, 5].

Subsumption. It turns out that in the presence of optional matching not only the semantics of subsumption is more robust than that of containment, but also its complexity is much more stable for the different fragments of SPARQL. In fact, the subsumption problem is Π_2^P-complete in all of the cases considered in this survey.

For CQs, the containment problem $Q_1 \subseteq Q_2$ is equivalent to deciding if there exists a homomorphism h from Q_2 to Q_1. Recall that the main intuition behind this is that h allows one to “translate” solutions of Q_1 to solutions of Q_2.

The subsumption problem for well-designed SPARQL queries can be decided in a similar way, and can essentially be reduced to a (possibly exponential) number of containment tests between CQs: An immediate consequence of Lemma 2 and Lemma 1 is that for every wdPF $(\mathcal{F}, \mathcal{X})$ and RDF graph G, every solution $\mu \in [\![(\mathcal{F}, \mathcal{X})]\!]_G$ is witnessed by some subtree $\mathcal{T}'$ of $\mathcal{F}$ and some extension μ' of μ s.t. (a) $\mathsf{dom}(\mu') = \mathsf{vars}(\mathcal{T}')$, (b) $\mathsf{dom}(\mu) = \mathsf{fvars}(\mathcal{T}')$, and (c) μ' maps all triple patterns in $\mathcal{T}'$ into G. Thus, subsumption between two wdPFs $(\mathcal{F}_1, \mathcal{X}), (\mathcal{F}_2, \mathcal{X})$ holds if and only if every such mapping μ' for $(\mathcal{F}_1, \mathcal{X})$ can be “translated” (in the same sense as for CQs) to a corresponding mapping on $(\mathcal{F}_2, \mathcal{X})$. This allows for the following characterization of subsumption between wdPFs.

Lemma 3 ([30]). *Let $(\mathcal{F}_1, \mathcal{X})$ and $(\mathcal{F}_2, \mathcal{X})$ be two wdPFs. Then $(\mathcal{F}_1, \mathcal{X}) \sqsubseteq (\mathcal{F}_2, \mathcal{X})$ iff for every subtree $\mathcal{T}_1'$ of $\mathcal{F}_1$, there exists a subtree $\mathcal{T}_2'$ of $\mathcal{F}_2$, s.t.*

(1) $\mathsf{fvars}(\mathcal{T}_1') \subseteq \mathsf{fvars}(\mathcal{T}_2')$ *and*
(2) there exists a homomorphism $h\colon pat(\mathcal{T}_2') \to pat(\mathcal{T}_1')$ with $h(?x) = ?x$ for all $?x \in \mathsf{fvars}(\mathcal{T}_1')$.

Assuming this characterization is satisfied, given some $\mu_1 \in [\![(\mathcal{F}_1, \mathcal{X})]\!]_G$ witnessed by some subtree $\mathcal{T}_1'$ and mapping μ_1', we get a corresponding mapping μ_2' as

Table 1. Complexity of the Containment and Equivalence problem, [25, 30].

	CONTAINMENT[S_1,S_2]				EQUIVALENCE[S_1,S_2]			
$\downarrow S_1$ / $S_2 \rightarrow$	$\emptyset$	$\{\cup\}$	$\{\pi\}$	$\{\cup,\pi\}$	$\emptyset$	$\{\cup\}$	$\{\pi\}$	$\{\cup,\pi\}$
$\emptyset$	NP-c.	Π_2^P-c.	undec.	undec.	NP-c.	–	–	–
$\{\cup\}$	NP-c.	Π_2^P-c.	undec.	undec.	Π_2^P-c.	Π_2^P-c.	–	–
$\{\pi\}$	NP-c.	Π_2^P-c.	undec.	undec.	Π_2^P-c.	Π_2^P-h.	undec.	–
$\{\cup,\pi\}$	NP-c.	Π_2^P-c.	undec.	undec.	Π_2^P-c.	undec.	undec.	undec.

$\mu_2'(\cdot) = \mu_1'(h(\cdot))$, where h is the homomorphism guaranteed to exist by the characterization. Observe, however, that the lemma only guarantees $\mu_2' \sqsubseteq [\![\mathcal{F}_2]\!]_G$, and not $\mu_2' \in [\![\mathcal{F}_2]\!]_G$, since μ_2' need not be maximal. Thus the characterization guarantees $\mu_2 \sqsubseteq [\![(\mathcal{F}_2, \mathcal{X})]\!]_G$ as required, but not $\mu_2 \in [\![(\mathcal{F}_2, \mathcal{X})]\!]_G$.

This characterization can be immediately turned into a Π_2^P-algorithm for deciding subsumption. On the other hand, Π_2^P-hardness was shown to already hold for SUBSUMPTION[$\emptyset$,$\emptyset$] in [25]. We thus get the following result.

Theorem 6 ([25, 30]). *The problem* SUBSUMPTION[S_1,S_2] *is Π_2^P-complete for all $S_1, S_2 \subseteq \{\cup, \pi\}$.*

Containment. The complexity of the CONTAINMENT problem is summarized in Table 1. Beside ranging from NP-completeness to even undecidability, it also displays a surprising asymmetry: For instance, CONTAINMENT[$\{\pi\}$,$\emptyset$] is NP-complete, while CONTAINMENT[$\emptyset$,$\{\pi\}$] is undecidable.

Recall that for subsumption $(\mathcal{F}_1, \mathcal{X}) \sqsubseteq (\mathcal{F}_2, \mathcal{X})$, the crucial property for the characterization in Lemma 3 to be correct is that it is irrelevant whether the subtrees of $\mathcal{F}_2$ are maximal or not. However, for containment, this is no longer the case since now it must be guaranteed that for every solution to $(\mathcal{F}_1, \mathcal{X})$, the exact same mapping (and not an extension of it) is also a solution to $(\mathcal{F}_2, \mathcal{X})$. While homomorphisms are too weak to directly express such a property, for the decidable cases in Table 1, it is possible to express this in an indirect way. We demonstrate this idea for the problem CONTAINMENT[$\{\pi\}$,$\emptyset$]:

Lemma 4 ([30]). *Let $(\mathcal{T}_1, \mathcal{X})$ and $\mathcal{T}_2$ be wdPTs. Then $(\mathcal{T}_1, \mathcal{X}) \subseteq \mathcal{T}_2$ iff for every subtree $\mathcal{T}_1'$ of $\mathcal{T}_1$,*

(1) either there exists a child node n of $\mathcal{T}_1'$ and a homomorphism $h\colon pat(n) \rightarrow pat(\mathcal{T}_1')$ with $h(?x) = ?x$ for all $?x \in \mathsf{vars}(n) \cap \mathsf{vars}(\mathcal{T}_1')$

(2) or there exists a subtree $\mathcal{T}_2'$ of $\mathcal{T}_2$, s.t. (a) $\mathsf{fvars}(\mathcal{T}_1') = \mathsf{vars}(\mathcal{T}_2')$, (b) $pat(\mathcal{T}_2') \subseteq pat(\mathcal{T}_1')$, and (c) for all extensions $\hat{\mathcal{T}_2}'$ of $\mathcal{T}_2'$ there exists an extension $\hat{\mathcal{T}}_1'$ of $\mathcal{T}_1'$ and a homomorphism $h\colon pat(\hat{\mathcal{T}}_1') \rightarrow pat(\mathcal{T}_1') \cup pat(\hat{\mathcal{T}}_2')$ with $h(?x) = ?x$ for all $?x \in \mathsf{vars}(\mathcal{T}_1')$.

The intuition of this characterization is as follows: Property (1) is a technical detail dealing with subtrees that can always be extended. This case was implicitly covered for subsumption but must now be made explicit. Property (2a) and (2b) are the adaptations of the properties (1) and (2) in Lemma 3: (2a) follows immediately from the fact that extensions are not allowed. For (2b), observe that in the present case, looking for a homomorphism as in property (2) in Lemma 3 means to look for a homomorphism that is the identity on all variables in its domain, hence degenerating to a subset inclusion test. Finally, property (2c) implicitly ensures that the mapping on $\mathcal{T}_2$ cannot be extended. Intuitively, it expresses the following: Assume some RDF graph G, a mapping $\mu \in [\![(\mathcal{T}_1, \mathcal{X})]\!]_G$ witnessed by $\mu_1 \in [\![\mathcal{T}_1]\!]_G$ and $\mathcal{T}_1'$. Assuming further the properties of the lemma to be satisfied, we know from (2a) and (2b) that $\mu \sqsubseteq [\![\mathcal{T}_2]\!]_G$. Thus assume to the contrary that $\mu \notin [\![\mathcal{T}_2]\!]_G$ because of some $\mu' \in [\![\mathcal{T}_2]\!]_G$ with $\mu \sqsubseteq \mu'$. Let $\mathcal{T}_2'$ be the subtree of $\mathcal{T}_2$ corresponding to μ and let μ' be witnessed by some subtree $\hat{\mathcal{T}}_2'$. Then $\hat{\mathcal{T}}_2'$ must be an extension of $\mathcal{T}_2'$. But then the subtree $\hat{\mathcal{T}}_1'$ of $\mathcal{T}_1$ and homomorphism h according to property (2c) provide a contradiction to $\mu_1 \in [\![\mathcal{T}_1]\!]_G$, since $\mu_1 \sqsubset \mu_1'$ where $\mu_1'(\cdot) = \mu'(h(\cdot))$, and $\mu_1' \in [\![\mathcal{T}_1]\!]_G$. I.e., the characterization guarantees the maximality on $\mathcal{T}_2$ implicitly by making sure that if the mapping is not maximal on $\mathcal{T}_2$, then it is not on $\mathcal{T}_1$ either. This characterization can be easily extended to Containment$[\{\cup, \pi\}, \emptyset]$.

A direct implementation of this characterization would lead to a Π_2^P-algorithm for deciding Containment$[\{\cup, \pi\}, \emptyset]$. However, it is in fact not necessary to perform the test for all subtrees of $\mathcal{T}_1$, but it suffices to just test a linear number of them. This pushes the complexity down to NP. Allowing for UNION on the right hand side requires some non-trivial extension of property (2c) for the characterization to still work in such a setting. As a result, the complexity rises by one level in the polynomial hierarchy.

Once projection is allowed to occur in the containing query (i.e. the query on the right hand side), this approach no longer works, and in fact the problem becomes undecidable. To get an idea of why this is the case, observe that an alternative way to look at property (2c) is that we create a canonical RDF graph over which μ – the mapping of interest – is guaranteed not to be a solution. Without projection, such a canonical graph can always be found since we get the following property: μ is not a solution if it can be extended to a bigger solution. It thus basically suffices to just add one such extension to the canonical graph. In the presence of projection, however, we get the following situation: μ is not a solution to $\mathcal{T}_2$, if for every subtree $\mathcal{T}_2'$ of $\mathcal{T}_2$ with $\mathsf{fvars}(\mathcal{T}_2') = \mathsf{dom}(\mu)$ *and every mapping* μ' *on* $\mathsf{evars}(\mathcal{T}_2')$, there exists an extension of $\mu \cup \mu'$ that is a solution to $\mathcal{T}_2$. Thus we have to provide an extension for all possible mappings μ' in the canonical graph. Adding these extensions may give rise to new mappings μ' on the existential variables, which in turn require new extensions to be provided in the canonical RDF graph, and it is not clear when this can be stopped.

This behavior is reminiscent of the chase termination (cf. [26, 16, 9] and related problems, and in fact the undecidability of the containment problem was shown by reduction from the following problem: Given a set Σ of tuple-generating dependencies, a database instance I and a Boolean CQ Q, is Q true in every (finite) model of Σ and I (cf. [9, 30])?

Equivalence. The complexity of the EQUIVALENCE problem is also depicted in Table 1. Of course, for two SPARQL queries Q_1 and Q_2 it holds that $Q_1 \equiv Q_2$ iff $Q_1 \subseteq Q_2$ and $Q_2 \subseteq Q_1$. Thus an upper-bound on the complexity of the containment problem also provides an upper-bound for the equivalence problem. In addition, it was shown in [25] that for queries in wd-SPARQL[$\emptyset$] it is also the case that $Q_1 \equiv Q_2$ iff $Q_1 \sqsubseteq Q_2$ and $Q_2 \sqsubseteq Q_1$. However, as soon as we add union or projection on either side, this property does no longer hold.

Of course, the EQUIVALENCE[S_1,S_2] problem is symmetric in S_1 and S_2. Hence, only the lower triangle of the table has been filled in. The reason that EQUIVALENCE[$\{\cup, \pi\}$,$\emptyset$] is decidable in Π_2^P, while CONTAINMENT[$\emptyset$,$\{\cup, \pi\}$] is undecidable is that in order to decide equivalence, it actually suffices to test containment in one, and only subsumption in the other direction. We would like to point out that not only is the exact complexity of EQUIVALENCE[$\{\pi\}$,$\{\cup\}$] still open, but it is even unknown if the problem is decidable or not.

5 Ontology-Based Query Answering

In the recently released recommendation [14], the W3C has defined various SPARQL entailment regimes to allow users to specify implicit knowledge about the vocabulary in an RDF graph. The theoretical underpinning of query answering under entailment regimes is provided by the big body of work on *ontology-based query answering*, notably in the area of description logics (DLs) [6]. However, the semantics of query answering under SPARQL entailment regimes is defined in a simpler and less expressive way than the *certain answer semantics* usually adopted in the DL and database literature.

Example 4. Consider an RDF graph G containing a single triple (“Star_Wars”, `rdf:type`, `movie`) – stating that “Star_Wars” is a movie – and an ontology $\mathcal{O}$ containing the triple (`movie`, `rdfs:subClassOf`, $\exists$`has_actor`). – stating that every movie has some actor who acts in it. Now consider the following simple graph pattern $(P, \{?x\})$ with $P = (?x, \texttt{has\_actor}, ?y)$, where $?x$ is the only output variable. Following the SPARQL entailment regimes standard [14], this query yields an empty result. ◇

By the concept inclusion (`movie`, `rdfs:subClassOf`, $\exists$`has_actor`), we know for certain that there is some actor who acts in “Star_Wars”. Hence, the result in the above example is rather unintuitive. The reason for this behavior is that the standard for SPARQL entailment regimes [14] requires that all values assigned to any variable must come from the RDF graph. In other words, distinguished variables (which are ultimately output) and non-distinguished variables (which are eventually projected out) are treated in the same way. In contrast, the certain answer semantics retrieves all mappings on the distinguished variables that allow to satisfy the query in every possible model of the database and the ontology – yielding the certain answer $\mu = \{?x \leftarrow \text{“Star\_Wars”}\}$ in the above example. The certain answer semantics has been extensively studied in the database and DL literature for CQs [1, 10]. However, in the presence of optional matching the usual certain answer semantics (i.e., something is a certain answer if it is present in every model) turns out to be unsatisfactory:

Example 5. Consider the graph pattern $(P, \{?x, ?z\})$ with P: $((?x, \texttt{has\_actor}, ?y)\mathsf{OPT}(?y, \texttt{was\_born}, ?z))$ over the graph $G = \{(\text{“Star\_Wars”}, \texttt{has\_actor}, \text{“Harrison\_Ford”})\}$ and empty ontology $\mathcal{O}$. The query yields a unique solution $\mu = \{?x \leftarrow \text{“Star\_Wars”}\}$. Clearly, also the extended graph $G' = G \cup \{(\text{“Harrison\_Ford”}, \texttt{was\_born}, \text{“1942”})\}$ is a model of $(G, \mathcal{O})$. But in G', μ is no longer a solution since μ can be extended to solution $\mu' = \{?x \leftarrow \text{“Star\_Wars”}, ?z \leftarrow \text{“1942”}\}$. Hence, there exists no mapping which is a solution in every possible model of $(G, \mathcal{O})$. $\diamond$

As Example 5 illustrates, a literal adoption of the certain answer semantics in the presence of the OPT operator leads to having no solutions, even though there is a solution that can be extended to a solution in every model. In order to tackle this and further problems, the definition of certain answers for the class of wdPTs has to be suitably modified [2]. This modification of the semantics also requires an adaptation and extension of known query answering algorithms established in the area of description logics. We mention two such modified algorithms for query evaluation under DL-Lite [10]. It turns out that the additional expressive power due to the certain answers comes without an increase of the complexity.

OWL 2 QL. RDF has been enhanced by the OWL 2 Web Ontology Language [27], a World Wide Web Consortium (W3C) recommendation to enable the specification of background knowledge about the application domain, and to enrich query answers with implicit information. The logical underpinning of OWL 2 and its sub-languages are description logics. One such sub-language is OWL 2 QL which is based on DL-Lite$_\mathcal{R}$, a member of the DL-lite family [10]. Its fundamental building blocks are *constants* c, *atomic concepts* A and *atomic roles* R, which are countably infinite and mutually disjoint subsets of a set $\mathbf{U}$ of URIs. From these we can build *basic roles* R and R^-, and *basic concepts* B and $\exists Q$, where Q is a basic role. Using the above, DL-Lite$_\mathcal{R}$ allows one to express the following kind of statements: Membership assertions $(c, \texttt{rdf:type}, B)$ or (c, Q, c'), concept inclusions $(B_1, \texttt{rdfs:subClassOf}, B_2)$, role inclusions $(Q_1, \texttt{rdfs:subPropertyOf}, Q_2)$ as well as concept and role disjointness (where c, c' are constants and B_i, Q_i are basic concepts resp. basic roles). In the following, an ontology $\mathcal{O}$ is any set of such expressions, excluding membership assertions, which we assume to be part of the RDF graph. A *knowledge base (KB)* $\mathcal{G} = (G, \mathcal{O})$ consists of an RDF graph G and an ontology $\mathcal{O}$.

Certain answers of wdPTs. Before providing our definition of certain answers, we need to introduce two additional notions. Let P be a well-designed graph pattern. Following [29], we say that P' is a reduction of P (denoted as $P' \trianglelefteq P$) if P' can be constructed from P by replacing in P sub-patterns of the form $(P_1\ \mathsf{OPT}\ P_2)$ by P_1. Note that, in terms of wdPTs, a reduction corresponds to a subtree containing the root node of the wdPT. Moreover, for a mapping μ and some property A, we shall say that μ is $\sqsubseteq$-*maximal w.r.t.* A if μ satisfies A, and there is no μ' such that $\mu \sqsubseteq \mu'$, $\mu' \not\sqsubseteq \mu$, and μ' satisfies A.

Definition 2. *Let $\mathcal{G} = (G, \mathcal{O})$ be a KB and $Q = (P, \mathcal{X})$ a well-designed graph pattern. A mapping μ is a* certain answer *to Q over $\mathcal{G}$ if it is a $\sqsubseteq$-maximal mapping with the following properties: (1) $\mu \sqsubseteq [\![Q]\!]_{G'}$ for every model G' of $\mathcal{G}$, and (2) $\mathsf{vars}(P') \cap \mathcal{X} = \mathsf{dom}(\mu)$ for some $P' \trianglelefteq P$. We denote by $\mathsf{cert}(P, \mathcal{X}, \mathcal{G})$ the set of all certain answers to Q over $\mathcal{G}$.*

The reason for restricting the set of certain answers to $\sqsubseteq$-maximal mappings is that queries with projection and/or UNION may have "subsumed" solutions, i.e. solutions s.t. also some proper extension is a solution. But then – with set semantics – we cannot recognize the reason why some subsumed solution may be not a solution in some possible world, as illustrated in Example 6.

Example 6. Let us revisit the graph pattern $(P, \{?x, ?z\})$ of Example 5 with $P = (?x, \texttt{has\_actor}, ?y)$. Consider the following RDF graph G:

$$\begin{aligned} G = \{ & (\text{"Star\_Wars"}, \texttt{has\_actor}, \text{"Harrison\_Ford"}), \\ & (\text{"Star\_Wars"}, \texttt{has\_actor}, \text{"Mark\_Hamill"}), \\ & (\text{"Harrison\_Ford"}, \texttt{was\_born}, \text{"1942"})\}. \end{aligned}$$

and empty ontology $\mathcal{O}$. As possible models of $(G, \mathcal{O})$ we have all graphs containing G. Hence, $\mu = \{?x \leftarrow \text{"Star\_Wars"}, ?z \leftarrow \text{"1942"}\}$ and $\mu' = \{?x \leftarrow \text{"Star\_Wars"}\}$ are both solutions of $(P, \{?x, ?z\})$ over G and can be extended to solutions in every possible model.

Next consider the RDF graph:

$$\begin{aligned} G' = \{ & (\text{"Star\_Wars"}, \texttt{has\_actor}, \text{"Harrison\_Ford"}), \\ & (\text{"Harrison\_Ford"}, \texttt{was\_born}, \text{"1942"})\}. \end{aligned}$$

If we take as certain answers all mappings that can be extended to some solution in every possible model, then μ' from above is still a certain answer, which is clearly undesired. ◇

A key idea to solve the problem illustrated in Example 6 is to allow only "maximal" solutions. In addition, Property (2) in Definition 2 ensures that the domain of such an answer adheres to the structure of nested OPT operators in the query. However, we can show that this property need not be considered during the computation of the certain answers, but can be enforced in a simple post-processing step. We call such answers that satisfy Definition 2 except Property (2) *certain pre-answers*, and use $\mathsf{certp}(P, \mathcal{X}, \mathcal{G})$ to denote the set of all certain pre-answers. The same is true for projection, which can also be performed in a simple post-processing step. Thus, it suffices to compute $\mathsf{certp}(P, \mathcal{G})$, which can be done via a universal solution (referred to as *canonical model* in the area of DLs) as follows.

Theorem 7 ([2]). *Let $\mathcal{G} = (G, \mathcal{O})$ be a KB and P a well-designed graph pattern. Then, $\mathsf{certp}(P, \mathcal{G}) = \mathrm{MAX}([\![P]\!]_{\mathsf{univ}(G)}\!\downarrow)$, where $\mathrm{MAX}(M)$ is the set of $\sqsubseteq$-maximal mappings in M, $M\!\downarrow := \{\mu\!\downarrow \mid \mu \in M\}$ ($\mu\!\downarrow$ is the restriction of μ to those variables which are mapped by μ to the active domain of G), and $\mathsf{univ}(G)$ is a universal solution of $\mathcal{G}$.*

However, computing the certain answers via a universal solution is not always practical, since universal solutions can be infinite. As a result, query rewriting algorithms have been developed. These algorithms take the input query and the ontology, and rewrite them into a single query that can be evaluated over the input database without considering the ontology. By introducing several adaptations and extensions of the rewriting-based CQ evaluation for DL-Lite from [10], we developed two different approaches to compute the certain answers for well-designed SPARQL graph patterns (or, equivalently, of wdPTs) under OWL 2 QL entailment [2].

The first one proceeds in a modular way by rewriting the pattern P_n at each node n in a wdPT individually. It thus follows the general philosophy of SPARQL entailment regimes [14]. One possible disadvantage of this modular approach is that it requires to maintain additional data structures to ensure consistency when combining the partial solutions for the patterns of different nodes. As a consequence, the complete algorithm has to be implemented from scratch because the standard tools cannot handle these additional data structures.

The goal of the second approach is to make use of standard technology as much as possible. The idea is to transform the OWL 2 QL entailment under our new semantics into SPARQL query evaluation under RDFS entailment [14], for which strong tools are available. Unlike the first – modular – approach, this rewriting proceeds in a holistic way, i.e. it always operates on the whole query.

Based on these rewriting algorithms, we can show that the complexity of query answering and of several static query analysis tasks does not increase despite the additional power of OWL 2 QL entailment under our new semantics.

Recall from Section 3 the two variants Partial-Eval and Max-Eval of the Evaluation problem of wdPTs, where we have to decide for a graph G, wdPT $(\mathcal{T}, \mathcal{X})$, and mapping μ, if μ can be extended to a solution or is a maximal solution, respectively, of $(\mathcal{T}, \mathcal{X})$ over G. Note that we cannot directly compare the Evaluation problem of wdPTs under OWL 2 QL entailment (with our certain answer semantics) and the Evaluation problem of wdPTs without entailment regimes. This is due to the fact that our certain answer semantics (for reasons explained above) only allows maximal solutions. Hence, Partial-Eval and Max-Eval are the right problems to look at. The Partial-Eval problem is NP-complete [25] and the Max-Eval problem is DP-complete [2], and it was shown in [2] that these complexities remain unchanged under OWL 2 QL entailment with our certain answer semantics.

As far as static query analysis is concerned, we now have to redefine the problems Subsumption, Containment, and Equivalence so as to take a given ontology into account. For instance, for the Subsumption problem, we are now given two wdPTs $\mathcal{T}_1, \mathcal{T}_2$ plus an ontology $\mathcal{O}$; we have to decide if for every RDF graph G, the relationship $\mathsf{cert}(\mathcal{T}_1, \mathcal{X}_1, \mathcal{G}) \sqsubseteq \mathsf{cert}(\mathcal{T}_2, \mathcal{X}_2, \mathcal{G})$ holds, where $\mathcal{G}$ denotes the knowledge base $\mathcal{G} = (G, \mathcal{O})$. Recall that Subsumption without entailment regimes is Π_2^P-complete in all settings considered here (i.e., with or without projection; with our without the UNION operator). It can be shown that the complexity remains the same also for the Subsumption problem under

OWL2 QL entailment. [2]. Note that the subsumption relation between two queries only depends on the *maximal* solutions over an arbitrary graph. Hence, we did not need to define yet another variant of SUBSUMPTION, which takes only the maximal solutions into account. In contrast, for CONTAINMENT and EQUIVALENCE, a comparison between the settings with and without ontologies only makes sense if we check for containment (resp. equivalence) of the maximal solutions only. In [2], the resulting problems were shown Π_2^P-complete both for the settings with and without OWL2 QL entailment.

6 Conclusion and Future Work

We have recalled some recent results on an interesting fragment of SPARQL, the so-called well-designed SPARQL graph patterns or, equivalently, well-designed pattern trees (wdPTs). Such queries can be seen as a natural extension of conjunctive queries (CQs) by the optional matching feature. It has turned out that this feature makes virtually all relevant computational tasks more complex: the complexity of the EVALUATION problem raises from NP-completeness to Σ_2^P-completeness. The CONTAINMENT and EQUIVALENCE problems even become undecidable unless we forbid projection. In [29], SUBSUMPTION has been proposed as an interesting variant of CONTAINMENT, which is computationally better behaved and which may be more intuitive in the presence of optional matching. Its complexity is Π_2^P-complete in all settings considered here. Finally, we have seen that an additional extension of wdPTs by entailment under OWL2 QL (which corresponds to DL-Lite$_\mathcal{R}$) does not increase the complexity anymore.

Note that many further aspects of well-designed SPARQL graph patterns have been studied, which were not recalled in this survey. In response to the intractabilty of the EVALUATION problem of wdPTs, works on the approximation of CQs [7] were extended to wdPTs in [8]. The COUNTING problem of wdPTs (i.e., given a wdPT $\mathcal{T}$ and a graph G, how many solutions does $\mathcal{T}$ have over G) was studied in [31]. The COUNTING problem turned out to be more complex than the EVALUATION problem in the sense that the restrictions guaranteeing tractability of EVALUATION do not suffice to achieve tractability of COUNTING. In [24], various aspects of the ENUMERATION problem of wdPTs (i.e., given a wdPT $\mathcal{T}$ and a graph G, output all solutions of $\mathcal{T}$ over graph G) are studied. As has been recalled in Section 3, tractability of the MAX-EVAL problem is easier to achieve than for the EVALUATION problem. Interestingly, for the ENUMERATION problem, outputting the *maximal* solutions may become harder than outputting *all* solutions.

Recall that projection is realized in SPARQL by wrapping a SPARQL graph pattern into a SELECT statement. Another query form provided by the SPARQL standard [33] is to wrap a SPARQL graph pattern into a CONSTRUCT statement. The result of applying a CONSTRUCT query to an RDF graph is again an RDF graph (rather than a set of mappings). In [23], several interesting properties of CONSTRUCT queries were presented. For instance, for CONSTRUCT queries with well-designed SPARQL graph patterns, it was shown that they correspond to positive first order queries. An important extension provided by SPARQL 1.1 [18] is the possibility to formulate queries which have to be evaluated over different endpoints. Various aspects of this federation extension were studied in [4] – including the extension of well-designed SPARQL to federated SPARQL queries. Most of the computational problems mentioned here (Evaluation, Containment, Equivalence, Subsumption) were studied in [22] for another important extension of SPARQL 1.1 [18] – the so-called property paths. This extension introduces the ability to navigate in RDF graphs. Property paths thus resemble regular path queries. However, as is shown in [22], the interaction with the other SPARQL operators – in particular, with the OPT operator – requires new techniques.

Despite the great variety of results obtained for wdPTs, many questions have remained open. First, in most of the complexity analyses carried out so far, some cases could not be fully classified. For instance, in Table 1, the exact complexity (even the question of decidability) of Equivalence$[\{\pi\},\{\cup\}]$ is still open. Closing these gaps is a natural task for future work. Strongly related to such complexity analyses is the quest for tractable fragments of the various problems studied so far. For instance, we have recalled here that the Evaluation problem of wdPTs becomes tractable if wdPTs are restricted to the class $\ell\text{-}\mathcal{C} \cap \mathsf{BI}(c)$ where $\mathcal{C}$ is $\mathsf{TW}(k)$ or $\mathsf{HW}(k)$ and $c \geq 1$. The same restriction guarantees tractability of the Enumeration problem [24], which is not the case for the Counting problem [31]. For all these problems, further approaches have to explored to find (further) "natural" tractable classes of wdPTs.

Finally, the language fragments studied so far should be extended in several directions. For instance, we have recalled here some results obtained for the evaluation of wdPTs under OWL2 QL entailment (or, equivalently, under DL-Lite$_{\mathcal{R}}$). This work should be extended to more expressive entailment regimes. Another important extension is concerned with extending well-designed SPARQL itself. We have recalled several favorable features of this fragment of SPARQL. For instance, the complexity of the Evaluation problem drops from PSPACE-completeness (for the AND/OPT-fragment without well-designedness restriction) to coNP (without projection) or Σ_2^P (with projection), respectively. However, the restriction to well-designedness may be too strong. Hence, very recently [19], the extension of well-designed SPARQL to *weakly well-designed* SPARQL has been presented, by allowing some typical uses of non-well-designedness. On the one hand, it is shown that the extension of well-designed SPARQL to weakly well-designed SPARQL does not make the Evaluation problem harder. On the other hand, the authors give evidence that the resulting fragment of SPARQL is practically highly relevant by observing that in DBpedia query logs, almost all queries containing the OPT operator are weakly well-designed. Of course, the study of the various computational tasks mentioned here should be extended to yet further (and bigger) fragments of SPARQL.

Acknowledgments. This work was supported by the Vienna Science and Technology Fund (WWTF), project ICT12-15 and by the Austrian Science Fund (FWF): P25207-N23 and W1255-N23.

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley, 1995.
2. S. Ahmetaj, W. Fischl, R. Pichler, M. Simkus, and S. Skritek. Towards reconciling SPARQL and certain answers. In *Proc. WWW 2015*, pages 23–33. ACM, 2015.
3. R. Angles and C. Gutierrez. The expressive power of SPARQL. In *Proc. ISWC 2008*, LNCS 5318, pages 114–129. Springer, 2008.
4. C. B. Aranda, M. Arenas, Ó. Corcho, and A. Polleres. Federating queries in SPARQL 1.1: Syntax, semantics and evaluation. *J. Web Sem.*, 18(1):1–17, 2013.
5. M. Arenas and J. Pérez. Querying semantic web data with SPARQL. In *Proc. PODS 2011*, pages 305–316. ACM, 2011.
6. F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
7. P. Barceló, L. Libkin, and M. Romero. Efficient approximations of conjunctive queries. *SIAM J. Comput.*, 43(3):1085–1130, 2014.
8. P. Barceló, R. Pichler, and S. Skritek. Efficient evaluation and approximation of well-designed pattern trees. In *Proc. PODS 2015*, pages 131–144. ACM, 2015.
9. A. Calì, G. Gottlob, and M. Kifer. Taming the infinite chase: Query answering under expressive relational constraints. In *Proc. KR 2008*, pages 70–80. AAAI Press, 2008.
10. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. Autom. Reasoning*, 39(3):385–429, 2007.
11. A. K. Chandra and P. M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proc. STOC 1977*, pages 77–90. ACM, 1977.
12. C. Chekuri and A. Rajaraman. Conjunctive query containment revisited. *Theor. Comput. Sci.*, 239(2):211–229, 2000.
13. R. Cyganiak, D. Wood, and M. Lanthaler. RDF 1.1 concepts and abstract syntax. W3C Recommendation, W3C, 2014. `http://www.w3.org/TR/rdf11-concepts`.
14. B. Glimm and C. Ogbuji. SPARQL 1.1 Entailment Regimes. W3C Recommendation, W3C, Mar. 2013. `http://www.w3.org/TR/sparql11-entailment`.
15. G. Gottlob, N. Leone, and F. Scarcello. Hypertree decompositions and tractable queries. *J. Comput. Syst. Sci.*, 64(3):579–627, 2002.
16. S. Greco, F. Spezzano, and I. Trubitsyna. Checking chase termination: Cyclicity analysis and rewriting techniques. *IEEE Trans. Knowl. Data Eng.*, 27(3):621–635, 2015.
17. M. Grohe and D. Marx. Constraint solving via fractional edge covers. *ACM Transactions on Algorithms*, 11(1):4, 2014.
18. S. Harris and A. Seaborne. SPARQL 1.1 Query Language. W3C Recommendation, W3C, Mar. 2013. `http://www.w3.org/TR/sparql11-query`.
19. M. Kaminski and E. V. Kostylev. Beyond well-designed SPARQL. In *Proc. ICDT 2016, to appear*, 2016.
20. Y. Kanza, W. Nutt, and Y. Sagiv. Querying incomplete information in semistructured data. *J. Comput. Syst. Sci.*, 64(3):655–693, 2002.
21. A. C. Klug. On conjunctive queries containing inequalities. *J. ACM*, 35(1):146–160, 1988.
22. E. V. Kostylev, J. L. Reutter, M. Romero, and D. Vrgoc. SPARQL with property paths. In *Proc. ISWC 2015*, LNCS 9366, pages 3–18. Springer, 2015.

23. E. V. Kostylev, J. L. Reutter, and M. Ugarte. CONSTRUCT Queries in SPARQL. In *Proc. ICDT 2015*, LIPIcs 31, pages 212–229, 2015.
24. M. Kröll, R. Pichler, and S. Skritek. On the complexity of enumerating the answers to well-designed pattern trees. In *Proc. ICDT 2016, to appear*, 2016.
25. A. Letelier, J. Pérez, R. Pichler, and S. Skritek. Static analysis and optimization of semantic web queries. *ACM Trans. Database Syst.*, 38(4):25, 2013.
26. M. Meier. *On the termination of the chase algorithm.* PhD thesis, University of Freiburg, 2010.
27. B. Motik, B. C. Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz. Owl 2 web ontology language: Profiles. W3c working draft, W3C, October 2008.
28. J. Pérez, M. Arenas, and C. Gutierrez. Semantics and complexity of SPARQL. In *Proc. ISWC 2006*, LNCS 4273, pages 30–43. Springer, 2006.
29. J. Pérez, M. Arenas, and C. Gutierrez. Semantics and complexity of SPARQL. *ACM Trans. Database Syst.*, 34(3), 2009.
30. R. Pichler and S. Skritek. Containment and equivalence of well-designed SPARQL. In *Proc. PODS 2014*, pages 39–50. ACM, 2014.
31. R. Pichler and S. Skritek. On the hardness of counting the solutions of SPARQL queries. In *Proc. AMW 2014*, CEUR Workshop Proceedings 1189. CEUR-WS.org, 2014.
32. A. Polleres. From SPARQl to rules (and back). In *Proc. WWW 2007*, pages 787–796. ACM, 2007.
33. E. Prud′hommeaux and A. Seaborne. SPARQL Query Language for RDF. W3C Recommendation, W3C, 2008. `http://www.w3.org/TR/rdf-sparql-query/`.
34. Y. Sagiv and M. Yannakakis. Equivalences among relational expressions with the union and difference operators. *J. ACM*, 27(4):633–655, 1980.
35. M. Schmidt, M. Meier, and G. Lausen. Foundations of SPARQL query optimization. In *Proc. ICDT 2010*, pages 4–33. ACM, 2010.
36. M. Yannakakis. Algorithms for acyclic database schemes. In *Proc. VLDB 1981*, pages 82–94. IEEE Computer Society, 1981.