Soundness and Completeness of SPARQL Query Containment Solver SpeCS

Mirko Spasića,b,* and Milena Vujošević Janičića

aFaculty of Mathematics University of Belgrade, Studentski Trg 16, Belgrade, Serbia
bOpenLink Software, London, United Kingdom

ARTICLE INFO

Keywords: SPARQL query containment FOL modeling SpeCS solver correctness soundness completeness

ABSTRACT

Tool SpeCS implements an efficient automated approach for reasoning about the SPARQL query containment problem. In this paper, we prove the correctness of this approach. We give precise semantics of the core subset of SPARQL language. We briefly discuss the procedure used for reducing the query containment problem into a formal logical framework. We prove that such reduction is both sound and complete for conjunctive queries, and also for some important cases of non-conjunctive queries containing operator union, operator optional, and subqueries. Soundness and completeness proofs are considered in both containment and subsumption forms.

1. Introduction

SPARQL (Simple Protocol and RDF Query Language) is a query language and data access protocol [47, 46] used for querying data in the form of Resource Description Framework (RDF) [31, 33, 6] within Semantic Web [30, 26, 27]. There is a large number of RDF data sources available [40]. For achieving practical usability of these large amounts of data, executions of SPARQL queries have to be highly optimized [56].

One of the central problems for SPARQL query optimizations is query containment, namely deciding if each result of one query is at the same time a result of another query despite of a dataset being queried, i.e. this property should hold for any given dataset [10, 13, 45, 15]. In the case of SPARQL query language, many authors consider a weaker form of the containment relation, i.e. subsumption [44, 35, 5, 45]. Instead of the strict requirement that each result of the first query should be also a result of the second one, for the subsumption relation it is enough that for each result of the first query, there is a result of the second query in which it can be embedded (subsumed), i.e. the result of the second query contains the same projections as in the result of the first query and eventually some additional ones. An example of SPARQL queries that are in the query containment relation and in the subsumption relation is given in Figure 1.

Many other important problems, like query equivalence and query satisfiability [13, 45, 15], can be reduced to query containment [56]. Additional applications of query containment solvers can be adopted from applications within relational databases [36, 20, 19, 23, 37, 9].

For being practically usable, it is important that a query containment solver is efficient, that it covers a wide range of the SPARQL language constructs and that it always gives correct answers. While efficiency and coverage of a solver can be experimentally assessed, e.g. by using the relevant benchmarks [16, 50], the correctness of an approach should be formally proved as experimental evaluation can only confirm the presence of correctness violations (and not the absence of correctness violations in general case [17]).

Correctness of an approach includes soundness and completeness. In the context of query containment, soundness means that the approach cannot prove that two input queries are in the containment relation if they are not. Completeness guarantees that for each pair of queries that are in the containment relation the approach will confirm that.

Correctness can be considered on two levels: procedure level and implementation level. Correctness of the procedure level guarantees that the proposed approach is correct. Unsound approaches are usually not useful and therefore soundness is considered as a basic requirement. For example, in the case of query containment, a trivial procedure that returns always true (stating for any two queries that these are in the containment relation) is not a sound one, because there exist two queries for which the procedure will claim the con-
tainment (as it always does), while they are not in the relation. On the other hand, completeness of this procedure is present as for any two queries satisfying the relation, the procedure will confirm that. Having completeness together with soundness is desirable, but it is usually achievable only on a subset of the considered problem. Correctness of the implementation level guarantees that the procedure is correctly implemented, i.e. that the software meets its specification.

Frameworks like Isabelle [42] and Coq [8], incorporate development of formally verified software. There are different solvers and tools developed within these frameworks [34, 38, 55]. However, such development is very expensive and time consuming. A step towards formally verified software is a formal correctness proof of the proposed procedure while correctness of the implementation is based on the standard software development techniques. There are several different approaches for deciding SPARQL query containment, and for some of them, correctness of their procedures is proved.

The most important SPARQL solvers that deal with query containment or query equivalence include AFMU [13], TS [22], SA [35], JASG [57], QCAN [48] and SpeCS [56]. AFMU solver uses an expressive modal logic i.e. μ-calculus, and reduces [14, 15] the containment problem to a satisfiability problem in a fragment of the μ-calculus without alternation [32]. It uses a tool for deciding satisfiability within μ-calculus [58]. TS solver also uses a variant of μ-calculus without alternation, specially designed for reasoning over finite trees [21]. For AFMU and TS, soundness and completeness of their procedures are proved [13, 21]. SA solver introduces an algebra of query plans for a subset of SPARQL graph patterns. The solver is a prototype used for theoretical examination of complexity of the query containment problem and its correctness is not considered. Similarly, JASG solver is based on algebra expression trees that are constructed after SPARQL queries and on a subgraph isomorphism solver. Correctness of its procedure is also not considered. On the other hand, QCAN is a tool (not a containment solver) for canonicalisation of monotone SPARQL queries, i.e. it reduces a query to an equivalent one in a canonical form. It can be used for checking the equivalence of queries in a syntactic way by comparing the string representations of their canonical forms. Soundness and completeness of this procedure is proved [48, 49].

SpeCS [56] is an open source query containment solver which is based on transforming the query containment problem into a satisfiability problem [7] in first order logic (FOL). SPARQL and FOL have the same expressive power [1] so reducing query containment into FOL satisfiability problem is a promising approach. According to thorough evaluation on the two most important benchmarks, Query Containment Benchmark [16] and SQCFramework [50], SpeCS covers all widely used SPARQL language constructs and it is highly efficient. Compared to other state-of-the-art solvers, SpeCS has a better performance and it supports a bigger number of language constructs [56].

In this paper, we prove the correctness of the approach used by SpeCS. We give the precise semantics of the covered SPARQL constructs and all definitions and proofs of lemmas that are necessary for proving correctness. All definitions are given recursively and proofs are done inductively, so when the approach is extended with additional language constructs, the definitions and proofs can also be incrementally extended.

Overview of the paper. Section 2 defines a relevant subset of SPARQL syntax and semantics, and formally defines problems of query containment, query subsumption, query equivalence and query satisfiability. Section 3 defines the signature of the used theory, the procedure for transforming conjunctive queries into FOL formulas and logical formulation of the containment and subsumption problem. It also gives proofs of lemmas that are connecting SPARQL terms, variables, expressions, conditions, graph patterns and queries with corresponding terms, variables, expressions, conditions and formulas in FOL. Section 4 gives necessary definitions and lemmas for proving correctness of the proposed modeling. It gives proofs of soundness and completeness of the proposed approach for both containment and subsumption problem. Section 5 deals with queries that are not conjunctive: queries that contain operators union and optional and that contain subqueries. Section 6 gives final conclusions and presents possible directions for further work.

2. SPARQL and Query Containment Problem

The Resource Description Framework (RDF), as a World Wide Web Consortium (W3C) recommendation, is an accepted data model for expressing information about World Wide Web resources in a flexible and extensible way. It is represented as a directed graph consisting of triple statements, i.e. RDF triples, where each of them is composed of a node for the subject, an edge for the predicate connecting the subject to an object, and a node for the object. Each of these three parts can be identified by an internationalised resource identifier - IRI (generalisation of the uniform resource identifiers) [18], subject and object can be unidentifed resources, i.e. blank nodes, while the object can also be a literal value. As we follow the standard definition of mutually disjoint, countable sets \( \forall \) (a set which contains variables), \( \exists \) (a set which contains IRIs), \( \mathbb{B} \) (a set which contains blank nodes) and \( \mathbb{L} \) (a set which contains literals) [24, 44, 15, 51, 3], an RDF triple is an element of the set \( \mathbb{B} \times \mathbb{I} \times \mathbb{L} \).

\[ I_{\mathbb{B}} \text{ denote an IRI and let } G_{\mathbb{L}} \text{ denote an RDF graph. Let } \text{different } \mathbb{RDF} \text{ graphs have disjoint sets of blank nodes. According to [56] we introduce the following definitions.} \]

**Definition 2.1 (RDF dataset)** An RDF dataset \( D \) is defined as a set containing a default graph named \( G_d \) and zero or more named graphs \( \langle i, G_i \rangle \):

\[
D \equiv \{ G_d, G_1, \ldots, G_n \}
\]

\[ \text{Like in [13], we abbreviate any union of sets } I, B, L \text{ and } V \text{ as, for instance, } \mathbb{B}L := I \cup \mathbb{B} \cup \mathbb{L}. \]

M. Spasić and M. Vujošević Janičić: Soundness and Completeness of SPARQL Query Containment Solver SpeCS
Definition 2.2 (Function $df$) Function $df$ maps a dataset $\mathcal{D}$ to its default graph:

$$df(\mathcal{D}) \equiv G_d$$

Definition 2.3 (Function $names$) Function $names$ maps a dataset $\mathcal{D}$ to the set of IRIs of its named graphs:

$$names(\mathcal{D}) \equiv \{i_1, \ldots, i_n\}$$

Definition 2.4 (Function $gr$) Function $gr$ maps a dataset and an IRI into a graph corresponding to the IRI within the given dataset:

$$gr_\mathcal{D}(i_k) \equiv \begin{cases} G_{i_k}, & \text{if } \langle i_k, G_{i_k} \rangle \in \mathcal{D} \\ G_\emptyset, & \text{otherwise}, \end{cases}$$

where $G_\emptyset$ is an empty graph.

Definition 2.5 (Function $merge$) Function $merge$ maps a subset $\{G_{i_1}, \ldots, G_{i_m}\}$ of RDF graphs into a graph containing a union of their nodes:

$$merge(\{G_{i_1}, \ldots, G_{i_m}\}) \equiv \bigcup_{i=1}^{m} G_{i_k}$$

2.1. SPARQL Syntax

SPARQL is a language for querying data stored in the relevant datasets, i.e. RDF graphs. The main feature of SPARQL logic used in the evaluation of queries is a pattern matching facility, i.e. a finding of relevant triple statements within a queried dataset based on a pattern specified by a query in question.

W3C precisely defined the SPARQL query grammar in the EBNF notation [47]. In this paper we discuss the simplified subset of this grammar, given in Figure 2, containing the most relevant language constructs used in practice.

Similar to other query languages, a SPARQL query contains:

- a select clause, specifying a set of variables of interest, i.e. distinguished variables, (described by nonterminal $Vars$ in the grammar). Instead of listing them, this clause can contain symbol $\ast$, meaning that the set of distinguished variables contains all the variables present in the graph pattern within where clause.

- (optionally) from and from named clauses for precise specification of the relevant dataset. Normally, a query is executed against an RDF dataset denoted by $\mathcal{D}$. Using these two clauses, a query defines a different dataset on which query evaluation is performed, i.e. query dataset (Definition 2.6).

- a where clause containing a graph pattern (described by nonterminal $GPatt$) for pattern matching within the query dataset.

2.2. SPARQL Semantics

The following definition specifies variables appearing in a language construct.

Definition 2.7 (Function $var$) Let $t$ be an RDF term, $tp$ be a triple pattern, $E$, $E_1$, $E_2$ expressions (described by nonterminal $Exp$ in the grammar), $R$, $R_1$, $R_2$ built-in conditions (described by nonterminal $Cond$ in the grammar), and $gp$, $gp_1$, $gp_2$ graph patterns. Variables appearing in the RDF term $t$, triple pattern $tp$, expression $E$, condition $R$ and graph pattern $gp$, in notation $var(t)$, $var(tp)$, $var(E)$, $var(R)$ and $var(gp)$ respectively, are

$$var(t) \equiv \begin{cases} \{t\}, & t \in VB \\ \emptyset, & t \in IL \end{cases}$$

Note that sets of blank nodes of different graphs $G_{i_k}$ are disjoint.
We follow the standard notation and set semantics of a SPARQL query and of a graph pattern evaluation [13, 43, 44, 1].

Intuitively, a result of a SPARQL query execution connects variables (and blank nodes which cannot be retrieved by the query) from the graph pattern of query to the values within a graph of the query dataset, i.e. to the values of IBL. Therefore, the function $\mu$ denotes a partial mapping from the set of variables and blank nodes $\text{VIB}$ to the set IBL. In the following text, $\text{dom}(m)$ denotes the domain of a mapping $m$, while the function $\mu_{\text{set}}$ denotes a mapping such that $\text{dom}(\mu_{\text{set}}) \ni \{x\}$ and $\mu_{\text{set}}(x) \ni c$.

**Definition 2.8** (Compatible mappings $\simeq$) Mappings $m_1$ and $m_2$ are compatible, denoted by $m_1 \simeq m_2$, if for each $x$ such that $x \in \text{dom}(m_1) \cap \text{dom}(m_2)$ it holds that $m_1(x) = m_2(x)$.

Note that two mappings with disjoint domains are always compatible. Also, if $m_1 \simeq m_2$, then $m_1 \cup m_2$ is also a mapping, such that it holds:

$$\text{dom}(m_1 \cup m_2) = \text{dom}(m_1) \cup \text{dom}(m_2)$$

$$(m_1 \cup m_2)(x) = \begin{cases} m_1(x), & x \in \text{dom}(m_1) \\ m_2(x), & \text{otherwise} \end{cases}$$

Furthermore, $m_1 \cup m_2$ is compatible with both $m_1$ and $m_2$.

**Definition 2.9** (Operations over sets of mappings) Let $\Omega_1$ and $\Omega_2$ be sets of mappings. Operations $\text{union}$, $\text{join}$, $\text{different}$...
ence, minus and left outer-join are defined as follows [44]:

\[\begin{align*}
\Omega_1 \cup \Omega_2 & \equiv \{ m \mid m \in \Omega_1 \text{ or } m \in \Omega_2 \}, \\
\Omega_1 \Join \Omega_2 & \equiv \{ m_1 \cup m_2 \mid m_1 \in \Omega_1, m_2 \in \Omega_2, m_1 \triangleq m_2 \}, \\
\Omega_1 \setminus \Omega_2 & \equiv \{ m_1 \mid m_1 \in \Omega_1 \text{ and for all } m_2 \in \Omega_2, m_1 \triangleq m_2 \text{ not compatible} \}, \\
\Omega_1 \setminus \Omega_2 & \equiv \{ m_1 \mid m_1 \in \Omega_1 \text{ and for all } m_2 \in \Omega_2, m_1 \triangleq m_2 \text{ are not compatible} \}, \\
\Omega_1 \bowtie \Omega_2 & \equiv (\Omega_1 \Join \Omega_2) \cap (\Omega_1 \setminus \Omega_2).
\end{align*}\]

Let \(\bar{x}\) denote a set containing variables \(x_i\), for each \(i \in \{1, \ldots, n\}\), i.e. \(\bar{x} = \{x_1, \ldots, x_n\}\).

**Definition 2.10** (Extension, restriction, projection) A mapping \(m_1\) is an **extension** of a mapping \(m_2\) (a mapping \(m_2\) is a **restriction** of a mapping \(m_1\)), denoted by \(m_1 \supseteq m_2\) (\(m_2 \subseteq m_1\)), if \(m_1 \supseteq m_2\) and \(\text{dom}(m_1) \supseteq \text{dom}(m_2)\). Projection of a mapping \(m\) to the set \(\bar{x}\), denoted by \(m_{\bar{x}}\), is a mapping such that \(\text{dom}(m_{\bar{x}}) = \text{dom}(m) \cap \bar{x}\) and \(m_{\bar{x}} \subseteq m\).

Note that if sets \(\text{dom}(m)\) and \(\bar{x}\) are disjoint, \(m_{\bar{x}}\) is an empty mapping, i.e. a mapping with an empty domain.

**Definition 2.11** (Projection operator) Let \(\Omega\) be a set of mappings, and \(\bar{x}\) be a set of variables. Then, a projection operator over the sets \(\bar{x}\) and \(\Omega\), denoted \(\Pi_{\bar{x}}(\Omega)\), is defined as:

\[\Pi_{\bar{x}}(\Omega) \equiv \{ m_{\bar{x}} \mid m \in \Omega \}\]

According to the SPARQL semantics [47], the following function is needed in Definition 2.13.

**Definition 2.12** (Semantics of function \(\text{dt}\)) Let \(1\) be a literal, function \(\text{dt}: \{1 \rightarrow \{\text{var}, \text{xsd:string}\}\}\) is defined as follows:

\[\text{dt}(1) \equiv \begin{cases} 
   \text{var}, & \text{if } 1 \text{ is typed literal, with } \sim_i \text{ in suffix, } i \in \{1, \ldots, n\}, \\
   \text{xsd:string}, & \text{if } 1 \text{ is untyped literal}.
\end{cases}\]

A mapping may be naturally extended on \(\text{IRIS}\), literals and variables and blank nodes outside of its domain, by introducing an additional constant named \(\text{err}\), as specified in the following definition.

**Definition 2.13** (Notation \([[:1]]_\mu\)) A value of an RDF term \(t\), an expression \(E\), and a triple pattern \(tp\), according to the mapping \(\mu\), in notation \([[:1]]_\mu\), \([[:E]]_\mu\), and \([[:tp]]_\mu\) respectively, is a value from \(\text{IBL}_e\), defined in the following way:

\[\begin{align*}
[[:t]]_\mu & \equiv \begin{cases} 
   t, & t \in \text{IL} \\
   \mu(t), & t \in \text{VB} \text{ and } t \in \text{dom}(\mu) \\
   \text{err}, & t \in \text{VB} \text{ and } t \notin \text{dom}(\mu)
\end{cases}, \\
[[:E]]_\mu & \equiv \begin{cases} 
   \text{dt}(\mu(E))_\mu, & \text{if } E \text{ is a variable or } E \text{ is a literal} \\
   \text{err}, & \text{else}
\end{cases}, \\
[[:tp]]_\mu & \equiv \begin{cases} 
   \text{tp is } s \text{ p o and } s \in \text{VIB}, \text{ p } \in \text{VI}, \text{ o } \in \text{VIBL}.
\end{cases}
\end{align*}\]

A set \(\text{IBL}_e\) is an extension of the set \(\text{IBL}\) containing this constant, i.e. \(\text{IBL}_e := \text{IBL} \cup \{\text{err}\}\).

**Definition 2.14** (Relation \(\text{IBL}\)) Let \(E_1\) and \(E_2\) be expressions, \(R_1\) and \(R_2\) built-in conditions. A mapping \(\mu\) satisfies built-in condition \(R\), denoted \(\mu \models R\), if:

\[\begin{align*}
[[:E]]_\mu & \notin \text{err}, \\
[[:E]]_\mu & \notin \text{err and } [[:E]]_\mu = [[:E]]_\mu, \text{ if } R \models E_1 = E_2 \\
\mu & \models R, \text{ if } R \models E
\end{align*}\]

**Definition 2.15** (Evaluation of a graph pattern \([[:1]]_\mu\)) Let \(G\) be an RDF dataset, \(G\) a graph within \(G\), \(tp\) a triple pattern, \(gp_1, gp_2\) graph patterns, and \(R\) a built-in condition. An evaluation of a graph pattern over the graph \(G\), which is in this context called the **active graph**, within the dataset \(D\), is defined recursively as follows:

\[\begin{align*}
[[:tp]]_G & \equiv \{ \mu \mid \text{dom}(\mu) = \text{var}(\text{tp}) \land [[:tp]]_\mu \subseteq G \}
\end{align*}\]

The following lemma connects a domain of an evaluation of a graph pattern and variables of a graph pattern.

**Lemma 2.1** Let \(D\) be an RDF dataset, \(G\) a graph within \(D\), \(gp\) a graph pattern which is union-free, optional-free and without subqueries, and \(\mu\) a mapping such that \(\mu \in \{gp\}_G\). Then:

\[\text{dom}(\mu) = \text{var}(gp)\].

**Proof.** The lemma is proved by induction over graph pattern \(gp\).

\[\text{dom}(\mu)^5\]

represents the evaluation of query \(q\_\mu\) over a dataset \(D\), introduced later in Definition 2.16.
By Definition 2.15, from \( \mu \in \mathcal{L}^{\text{gp}_1, \text{gp}_2}_c \), it holds
\[
\mu \in \mathcal{L}^{\text{gp}_1}_c \bowtie \mathcal{L}^{\text{gp}_2}_c.
\]

Therefore, by Definition 2.9, there exist compatible mappings \( \mu_1 \) and \( \mu_2 \), such that \( \mu = \mu_1 \cup \mu_2 \).
\[
\mu_1 \in \mathcal{L}^{\text{gp}_1}_c \text{ and } \mu_2 \in \mathcal{L}^{\text{gp}_2}_c.
\]

Therefore, it holds \( \text{dom}(\mu) = \text{dom}(\mu_1) \cup \text{dom}(\mu_2) \). By induction hypothesis, it holds
\[
\text{dom}(\mu_1) = \text{var}(\text{gp}_1) \text{ and } \text{dom}(\mu_2) = \text{var}(\text{gp}_2).
\]
Then, \( \text{dom}(\mu) = \text{var}(\text{gp}_1) \cup \text{var}(\text{gp}_2) \), i.e. by Definition 2.7, it holds \( \text{dom}(\mu) = \text{var}(\text{gp}_1, \text{gp}_2) \).

By Definition 2.15, from \( \mu \in \mathcal{L}^{\text{gp}_1, \text{filter}_R}_c \), it holds
\[
\mu \in \mathcal{L}^{\text{gp}_1}_c. 
\]
By induction hypothesis, it holds \( \text{dom}(\mu) = \text{var}(\text{gp}_1) \). Therefore, by Definition 2.7, it holds \( \text{dom}(\mu) = \text{var}(\text{gp}_1, \text{filter}_R) \).

By Definition 2.15, from \( \mu \in \mathcal{L}^{\text{gp}_1}_c \), it holds
\[
\mu \in \mathcal{L}^{\text{gp}_1}_c. 
\]
By induction hypothesis, it holds \( \text{dom}(\mu) = \text{var}(\text{gp}_1) \). Therefore, by Definition 2.7, it holds \( \text{dom}(\mu) = \text{var}(\text{gp}_1) \).

By Definition 2.15, from \( \mu \in \mathcal{L}^{\text{gp}_1}_c \), it holds
\[
\mu \in \mathcal{L}^{\text{gp}_1}_c \setminus \mathcal{L}^{\text{gp}_2}_c.
\]
Therefore, by Definition 2.9, it holds \( \mu \in \mathcal{L}^{\text{gp}_1}_c \). By induction hypothesis, it holds \( \text{dom}(\mu) = \text{var}(\text{gp}_1) \). Then, by Definition 2.7, it holds \( \text{dom}(\mu) = \text{var}(\text{gp}_1, \text{diff}_{\text{gp}_2}) \).

By Definition 2.15, from \( \mu \in \mathcal{L}^{\text{gp}_1}_c \), it holds
\[
\mu \in \mathcal{L}^{\text{gp}_1}_c \setminus \mathcal{L}^{\text{gp}_2}_c.
\]
Therefore, by Definition 2.9, it holds \( \mu \in \mathcal{L}^{\text{gp}_1}_c \). By induction hypothesis, it holds \( \text{dom}(\mu) = \text{var}(\text{gp}_1) \). Then, by Definition 2.7, it holds \( \text{dom}(\mu) = \text{var}(\text{gp}_1, \text{minus}_{\text{gp}_2}) \).

By Definition 2.15, from \( \mu \in \mathcal{L}^{\text{gp}_1}_c \), it holds
\[
\mu \in \mathcal{L}^{\text{gp}_1}_c \setminus \mathcal{L}^{\text{gp}_2}_c.
\]
Therefore, by Definition 2.9, it holds \( \mu \in \mathcal{L}^{\text{gp}_1}_c \). By induction hypothesis, it holds \( \text{dom}(\mu) = \text{var}(\text{gp}_1) \). Then, by Definition 2.7, it holds \( \text{dom}(\mu) = \text{var}(\text{gp}_1, \text{minus}_{\text{gp}_2}) \).

By Definition 2.15, from
\[
\mu \in \mathcal{L}^{\text{graph} \times \{\text{gp}_1\}_c, 
\]
it holds
\[
\mu \in \bigcup_{i \in \text{names}(\mathcal{D})} \left( \mathcal{L}^{\text{gp}_1}_c \bowtie \{\mu_{x \rightarrow i}\} \right).
\]

Therefore, by Definition 2.9 and induction hypothesis, it holds
\[
\text{dom}(\mu) = \text{var}(\text{gp}_1) \cup \{x\}.
\]
Then, by Definition 2.7, it holds
\[
\text{dom}(\mu) = \text{var}(\text{graph} \times \{\text{gp}_1\})
\]

By Definition 2.15, from
\[
\mu \in \mathcal{L}^{\text{graph} \times \{\text{gp}_1\}}_c,
\]
it holds
\[
\mu \in \bigcup_{i \in \text{names}(\mathcal{D})} \left( \mathcal{L}^{\text{gp}_1}_c \bowtie \{\mu_{x \rightarrow i}\} \right).
\]

\[\square\]

**Definition 2.16** (Evaluation of a query \([q]^D\)) An evaluation of a query \(q\) over a dataset \(D\) is defined as
\[
[q]^D \equiv \Pi_{\text{gp}}(\text{qpat})_{D[0]}^D
\]
where within the query \(q\), \(\text{qpat}\) is a query pattern, \(\overline{\nu}\) is a set of distinguished variables, and \(\mathcal{D}\) is a dataset.

In a case when a mapping \(\mu\) from \([\text{qpat}]^D_{\mathcal{D}[0]}\) has a disjoint domain with \(\overline{\nu}\), \([q]^D\) contains the empty mapping, corresponding to the empty SPARQL solution. In a case when the set \([q]^D\) is an empty set, there is no result of a query evaluation.

Note that each query specifies the active graphs that are used for evaluation of its graph patterns. For each considered graph pattern (Figure 2) the active graph is equal to the default graph of the query dataset, except in case for \(\text{graph} \times \{\text{gp}\}\) and \(\text{graph} \in \{\text{gp}\}\). These two constructs give a possibility to match the graph pattern \(\text{gp}\) against named graphs in the query dataset [1].

**Definition 2.17** (Relevant variables \(\overline{\nu}\)) For an RDF dataset \(\mathcal{D}\), a query \(q\) and a mapping \(\mu\) such that \(\mu \in [q]^D\), all variables from \(\text{dom}(\mu)\) are called relevant variables.

There is a slight but important difference between relevant and distinguished variables. Although every relevant variable is at the same time a distinguished variable, in the opposite direction this does not hold. A projection variable appearing only in the select clause and nowhere else in the query pattern is an example of a distinguished variable that is not relevant. The following lemma connects the relevant variables with the corresponding query pattern and distinguished variables.
Lemma 2.2 Let \( q \) be a query, \( \bar{dv} \) a set of its distinguished variables, \( q\text{pat} \) its graph pattern which is union-free, optional-free and without subqueries, and \( \bar{f}v \) a set of its relevant variables. Then:

\[
\bar{f}v = \text{var}(q\text{pat}) \cap \bar{dv}.
\]

Proof. By Definition 2.17, there is a mapping \( \mu \) and a dataset \( D \) such that \( \mu \in [q]^D \) and \( \bar{f}v = \text{dom}(\mu) \). Then, by Definition 2.16, it holds

\[
\mu \in \Pi_{\bar{f}v}([q\text{pat}]^D_{df(q)}).
\]

By Definition 2.11, there exists \( \mu' \) such that \( \mu \subseteq_{\bar{dv}} \mu' \) and

\[
\mu' \in [q\text{pat}]^D_{df(q)}.
\]

By Lemma 2.1, it holds \( \text{dom}(\mu') = \text{var}(q\text{pat}) \). Therefore, by Definition 2.10, it holds

\[
\text{dom}(\mu) = \text{dom}(\mu') \cap \bar{dv} = \text{var}(q\text{pat}) \cap \bar{dv},
\]

i.e.,

\[
\bar{f}v = \text{var}(q\text{pat}) \cap \bar{dv}.
\]

\[\square\]

2.3. Containment, subsumption, equivalence and satisfiability

Definition 2.18 (Containment \( q_1 \sqsubseteq q_2 \)). Given two queries \( q_1 \) and \( q_2 \), \( q_1 \) is contained in \( q_2 \) if, for every RDF dataset \( D \), it holds \([q_1]^D \subseteq [q_2]^D \).

If \( q_1 \) is contained in \( q_2 \), then \( q_2 \) is called a super-query, and \( q_1 \) is called a sub-query. The query containment problem is a problem to determine whether \( q_1 \) is contained in \( q_2 \). The problem is undecidable if query \( q_2 \) contains projections [45]. Therefore, we assume that all variables in the graph pattern of \( q_2 \) appear in the select clause.

Definition 2.19 (Subsumption \( q_1 \sqsubseteq q_2 \)). Given two queries \( q_1 \) and \( q_2 \), \( q_1 \) is subsumed by \( q_2 \), if for every RDF dataset \( D \), for each mapping \( \mu \) from \([q_1]^D \) there exists an extension \( \mu' \geq \mu \), such that \( \mu' \) belongs to \([q_2]^D \).

Subsumption relation can be considered as a weaker form of containment [44, 35, 5, 45], as \( q_1 \sqsubseteq q_2 \) implies \( q_1 \sqsubseteq q_2 \) (a proof uses Definition 2.18 and an extension \( \mu' = \mu \)). The same terminology concerning \( q_2 \) as a super-query and \( q_1 \) as a sub-query is used when subsumption relation is considered instead of query containment relation. The subsumption problem is a problem to determine whether \( q_1 \) is subsumed in \( q_2 \). Unlike query containment problem, query subsumption problem is not undecidable when query \( q_2 \) contains projections [45].

Definition 2.20 (Equivalence \( q_1 \equiv q_2 \)). Given two queries \( q_1 \) and \( q_2 \), \( q_1 \) is equivalent to \( q_2 \), if for every RDF dataset \( D \), it holds \([q_1]^D = [q_2]^D \).

Query equivalence corresponds to satisfying containment relation in both directions. But, if two queries satisfy the subsumption relation in both directions, they do not have to be equivalent (if either of them contains union or projection operator) [45].

Definition 2.21 (Query satisfiability, unsatisfiability) Query \( q \) is satisfiable if there exist a dataset \( D \) and a mapping \( \mu \) such that \( \mu \in [q]^D \). Otherwise, \( q \) is unsatisfiable.

For each query it holds that an unsatisfiable query is its subquery.

3. Modeling the QC Problem of Conjunctive Queries

In this section, we model a subset of SPARQL queries presented by the grammar in Figure 2, i.e. conjunctive queries [10, 12] extended by SPARQL negation, the operator graph and built-in functions, while the rest of language constructs are covered in Section 5. We translate such queries into FOL formulas (Section 3.2) that are used for reasoning about query relations (Section 3.3). We have already presented the modeling in [56], while here, we reshow the definitions that are necessary to prove the correctness of our reduction.

3.1. Theory signature

The theory signature used for reasoning about queries \( q_1 \) and \( q_2 \), denoted by \( L \), is given in Definition 3.2 (Figure 3) as a tuple

\[(\text{C} \cup \text{P}, \text{P} \cup \{\beta_d, \beta_n\}, \text{ar})\]

For example, for queries presented in Figure 1, the set \( \text{C} \) contains only constants \( \text{LosAngeles} \) (corresponding to the literal “Los Angeles”) and \( \text{a}, \text{Album}, \text{artist}, \text{SoloArtist}, \text{hometown}, \text{and name} \) (corresponding to the IRIs \( \text{a}, \text{:Album}, \text{:artist}, \text{:SoloArtist}, \text{:hometown} \) and \( \text{:name} \)). For the same pair of queries, the set \( \text{P} \) is an empty set, while the set \( \text{P} \) contains only the equality predicate symbol. Considering the grammar presented in Figure 2, the set \( \text{F} \) can contain the symbol \( \text{datatype} \), corresponding to the SPARQL function \( \text{datatype} \) while the set \( \text{P} \) can also have the symbol \( \text{isliteral} \), corresponding to the SPARQL predicate \( \text{isliteral} \), in cases where these built-in functions are used within the relevant queries. Predicate symbols \( \beta_d \) and \( \beta_n \) model belonging of a triple to the graphs of an RDF dataset \( D \), e.g.:

- \( \beta_d(x, a, \text{Album}) \) models that the triple \( ?x a :\text{Album} \) belongs to the default graph of an RDF dataset \( D \),
- \( \beta_n(x, a, \text{Album}, g) \) models that the triple \( ?x a :\text{Album} \) belongs to the named graph specified by the IRI \( g \) of an RDF dataset \( D \), where \( g \) is a corresponding constant to the IRI \( g \).

A set of variables \( \text{V} \) corresponds to the variables and blank nodes from the SPARQL queries \( q_1 \) and \( q_2 \). For the query pair presented in Figure 1, \( \text{V} \) is equal to a set containing variables \( x, v, z_i, \) and \( w_i \), corresponding to the SPARQL variables \( ?x \), \( ?y \), \( ?z \) and \( ?w \), respectively.

3.2. Transforming Conjunctive Queries Into Formulas

The following auxiliary definition extends a function to a subset of its domain.
Definition 3.1 (Function over sets) Let \( A \) be a set of elements, and \( f \) a function such that \( A \subseteq \text{dom}(f) \). Function \( f \) over set \( A \), in notation \( f(A) \), denotes the set \( \{ f(e) \mid e \in A \} \).

A translation of the SPARQL queries into the corresponding FOL formulas is performed by function \( \sigma \) recursively, SPARQL construct by construct, using the auxiliary functions \( \sigma_1 \) and \( cx \). Their definitions are given in Figure 3.

Function \( \sigma_1 \) assigns a FOL constant to a SPARQL IRI or a literal, and a FOL variable to a SPARQL variable or a blank node.

Function \( cx \) maps an active graph to the set of corresponding graph IRI constants.\(^6\) Note that, according to the semantics, the active graph \( G \) can be the default graph or a named graph of a query dataset \( \emptyset \). If \( G \) is the default graph, it can be equal to the default graph of \( D \), an RDF merge of one or more graphs, or an empty graph \( \emptyset \). These possibilities correspond to the cases present in Definition 3.4.

Definition 3.5 presents a translation of the SPARQL terms, expressions, conditions and graph patterns into the corresponding FOL formulas from the signature \( \mathcal{L} \). The full notation of the function \( \sigma \) includes an active graph \( G \) used for matching the pattern within the query. For readability reasons, the active graph is denoted in superscript, i.e. \( \sigma^G \), and is omitted when it is obvious from the context and for terms, expressions and conditions as these do not depend on the active graph. Note that the active graph can be changed only in case of the operator graph.

Definition 3.6 specifies formula \( \Phi(\mathcal{T}) \) corresponding to a query \( \mathcal{T} \), with a graph pattern \( \mathcal{Q} \) specifying a query dataset \( \emptyset \). As a parameter, this formula has a set \( \overline{\mathcal{V}} \) of some variables from \( \mathcal{V} \), while \( \overline{\mathcal{O}} \) denote other free variables in the formula \( \sigma(\mathcal{Q}) \). If distinguished variables are used as a parameter \( \overline{\mathcal{V}} \), then, intuitively, variables \( \overline{\mathcal{O}} \) within formula \( \Phi(\overline{\mathcal{V}}) \) denote variables form \( \mathcal{V} \) that correspond to variables from \( \mathcal{V} \) that are used in the query \( \mathcal{T} \) but that are not selected.

Definition 3.7 (Function \( \text{var} \) over formulas) Let \( \Phi \) be a FOL formula. \( \text{var}(\Phi) \) denotes a set of free variables appearing in the formula \( \Phi \).

The following four lemmas connect functions \( \text{var}, \sigma \) and \( \text{var} \).

**Lemma 3.1** Let \( t \) be a term. Then:

\[ \text{var}(\sigma(t)) = \sigma(\text{var}(t)). \]

*Proof.* The specified equality is proved by induction over a term \( t \).

\[
\begin{align*}
\text{var}(\sigma(c)) & = \text{var}(\sigma_1(c)) \\
(\text{by Def 3.5}) & = \text{var}(\sigma(c)) \\
(\text{by Def 3.3}) & = \emptyset \\
(\text{by Def 3.7}) & = \emptyset \\
(\text{by Def 3.1}) & = \sigma(\emptyset) \\
(\text{by Def 2.7}) & = \sigma(\text{var}(c)) \\
\end{align*}
\]

**Lemma 3.2** Let \( \varepsilon \) be an expression. Then:

\[ \text{var}(\sigma(\varepsilon)) = \sigma(\text{var}(\varepsilon)). \]

*Proof.* The specified equality is proved by induction over an expression \( \varepsilon \).

\[
\begin{align*}
\text{if} \quad \varepsilon & = t \\
\text{then} \quad \text{var}(\sigma(t)) & = \sigma(\text{var}(t)) \\
(\text{by Lemma 3.1}) & = \sigma(\text{var}(t)) \\
\text{if} \quad \varepsilon & = \text{datatype}(\mathcal{E}_1) \\
\text{then} \quad \text{var}(\sigma(\text{datatype}(\mathcal{E}_1))) & = \text{var}(\text{datatype}(\sigma(\mathcal{E}_1))) \\
(\text{by Def 3.5}) & = \text{var}(\text{ datatype}(\sigma(\mathcal{E}_1))) \\
(\text{by Def 3.7}) & = \text{var}(\sigma(\mathcal{E}_1)) \\
(\text{by induc.hyp.}) & = \sigma(\text{var}(\mathcal{E}_1)) \\
(\text{by Def 2.7}) & = \sigma(\text{var}(\text{ datatype}(\mathcal{E}_1)))) \\
\end{align*}
\]

**Lemma 3.3** Let \( \mathcal{R} \) be a condition. Then:

\[ \text{var}(\sigma(\mathcal{R})) = \sigma(\text{var}(\mathcal{R})). \]

*Proof.* The specified equality is proved by induction over a condition \( \mathcal{R} \).

\[
\begin{align*}
\text{if} \quad \mathcal{R} & = \mathcal{E}_1 = \mathcal{E}_2 \\
\text{then} \quad \text{var}(\sigma(\mathcal{E}_1 = \mathcal{E}_2)) & = \text{var}(\sigma(\mathcal{E}_1)) = \sigma(\mathcal{E}_2) \\
(\text{by Def 3.5}) & = \text{var}(\sigma(\mathcal{E}_1)) = \sigma(\mathcal{E}_2) \\
(\text{by Def 3.7}) & = \text{var}(\sigma(\mathcal{E}_1)) = \sigma(\mathcal{E}_2) \\
(\text{by Lemma 3.2}) & = \text{var}(\sigma(\mathcal{E}_1)) = \sigma(\mathcal{E}_2) \\
(\text{by Def 3.1}) & = \text{var}(\sigma(\text{var}(\mathcal{E}_1 = \mathcal{E}_2))) \\
(\text{by Def 2.7}) & = \sigma(\text{var}(\mathcal{E}_1 = \mathcal{E}_2)) \\
\text{if} \quad \mathcal{R} & = \Diamond \mathcal{R}_1 \\
\text{then} \quad \text{var}(\sigma(\Diamond \mathcal{R}_1)) & = \text{var}(\sigma(\Diamond \mathcal{R}_1)) \\
(\text{by Def 3.5}) & = \text{var}(\sigma(\Diamond \mathcal{R}_1)) \\
(\text{by Def 3.7}) & = \text{var}(\sigma(\Diamond \mathcal{R}_1)) \\
(\text{by induc.hyp.}) & = \sigma(\text{var}(\mathcal{R}_1)) \\
(\text{by Def 2.7}) & = \sigma(\text{var}(\Diamond \mathcal{R}_1)) \\
\end{align*}
\]

---

\(^6\)The function \( cx \) is not defined on the default graph of the RDF dataset \( D \) as there is no IRI associated to it.
Definition 3.2 (Theory signature $\mathcal{L}$ corresponding to the queries $Q_1$ and $Q_2$)

$F_j := C \cup F$ is a set of function symbols, where:
- $C$ is a set of constants (function symbols with arity 0) corresponding to the literals and IRIs that appear in the queries, and a constant $err$, corresponding to $err$ in $F$.
- $F$ is a set of function symbols corresponding to the built-in SPARQL functions used in the queries $Q_1$ and $Q_2$.

$P_j := P \cup \{\beta_1, \beta_2\}$ is a set of predicate symbols, where:
- $P$ is a set which contains the equality predicate symbol ($\equiv$) and other predicate symbols corresponding to the boolean SPARQL functions used in $Q_1$ and $Q_2$.
- $\beta_1$ and $\beta_2$ are predicate symbols that intuitively model belonging of a triple to the default graph of an RDF dataset $D$, or to a named graph specified by an IRI, respectively.
- $ar$ is an arity function:

\[
ar(a) := \begin{cases} 
0, & \text{if } a \in C, \\
1, & \text{if } a \in P \text{ and } a \text{ is datatype,} \\
1, & \text{if } a \in P \text{ and } a \text{ is isliteral}, \\
4, & \text{if } a \in P, \text{ and } a \text{ is $\beta_1$,} \\
4, & \text{if } a \in P, \text{ and } a \text{ is $\beta_2$.}
\end{cases}
\]

Definition 3.3 (Function $\sigma_i : \text{VIBLe} \to \mathcal{Y} \cup \mathcal{C}$)

\[
\sigma_i(t) := \begin{cases} 
\{c (c \in C), & \text{if } t = c \text{ and } c \in \text{all,} \\
v (v \in \mathcal{Y}), & \text{if } t = v \text{ and } v \in \text{all,} \\
err, & \text{if } t = err.
\end{cases}
\]

Definition 3.4 (Function $cx$)

\[
cx(g) := \begin{cases} 
\{s_i(k_1, \ldots, k_m) \mid g(\Gamma_\text{in}(f_1)) \subseteq D\}, & \text{if } g \text{ is the default graph of the query dataset } D \\
\{s_i(), \text{merge}(s_1, \ldots, s_n) \mid s_i(f_1) = g, j \in \{1, \ldots, m\}\}, & \text{if } g \text{ is the named graph of the query dataset } D
\end{cases}
\]

Definition 3.5 (Function $\sigma$)

\[
\sigma(t) := \sigma_i(t), \quad t \in \text{VIBL}
\]

Definition 3.6 (Formula $\Phi(\mathcal{V})$ corresponding to $Q$)

\[
\Phi(\mathcal{V}) := \exists \mathcal{V} \sigma(\mathcal{V})(\mathcal{q})
\]

Figure 3: Definitions taken from [56] are given here in a short form: theory signature $\mathcal{L}$ corresponding to the queries $Q_1$ and $Q_2$, functions $\sigma_i$, $cx$ and $\sigma$ for transforming queries into formulas and formula $\Phi(\mathcal{V})$ corresponding to the query $Q$. $E_1$ and $E_2$ are used for expressions, $R_1$ and $R_2$ for conditions, $s$, $p$ and $o$ for subjects, predicates and objects, respectively ($s \in \text{VIBL}, p \in \text{VIBL}, o \in \text{VIBL}$), $gp_1$ and $gp_2$ for graph patterns, $x$ for a SPARQL variable, $i$ for an IRI, while $I_n$ is a set containing all graph IRIs corresponding to the named graphs of the query dataset $D$. $\mathcal{V}$ stands for $\mathcal{V}(\mathcal{GP}(p_2)) \cup \mathcal{V}(\mathcal{GP}(p_1))$. 

\[
\begin{align*}
\text{R is } R_1 & \land R_2 \\
\text{var}(\sigma(R_1 \land R_2)) & = \sigma(\text{var}(R_1) \cup \text{var}(R_2)) \quad \text{(by Def 3.1)} \\
\text{by Def 3.5) & = \text{var}(\sigma(R_1) \land \sigma(R_2)) \quad \text{R is } R_1 \mid R_2 \quad \text{(by Def 2.7)} \\
\text{by Def 3.7) & = \text{var}(\sigma(R_1)) \cup \text{var}(\sigma(R_2)) \quad \text{R is } R_1 \mid R_2 \quad \text{Proof is analogous to the previous case.} \\
\text{by induc. hyp.)} & = \sigma(\text{var}(R_1) \cup \text{var}(R_2)) \quad \text{R is } R_1
\end{align*}
\]
Lemma 3.4 Let $gp$ be a graph pattern. Then, if all active graphs that are used for matching $gp$ are nonempty, it holds:

$$\text{var}(\sigma(gp)) = \text{var}(\text{var}(gp)).$$

Proof. The specified equality is proved by induction over a graph pattern $gp$.

If $g \notin \text{dom}(cx)$:

$$\text{var}(\sigma^*(s p o)) =$$

(by Def 3.5) $$= \text{var}(\beta_d(s, p, o))$$

(by Def 3.7) $$= \{s, p, o\} \cap \varnothing$$

(by Def 3.3) $$= \{s, p, o\} \cap \varnothing$$

(by Def 3.5) $$= \{s, p, o\} \cap \varnothing$$

(by Def 3.1) $$= \{s, p, o\} \cap \varnothing$$

(by Def of \text{U}) $$= \varnothing$$

(by dist. of \text{U}) $$= \varnothing$$

(by Def 2.7) $$= \text{var}(\text{var}(s p o))$$

(by Def 2.7) $$= \text{var}(s p o)$$

If $g \in \text{dom}(cx)$, as graph $g$ is a nonempty graph, by Definition 3.4, it holds $cx(g) \neq \varnothing$.

$$\text{var}(\sigma^*(s p o)) =$$

(by Def 3.5) $$= \text{var}(\beta_d(s, p, o))$$

(by Def 3.7) $$= \bigcup_{i \in \text{ctx}(g)} \text{var}(\beta_d(s, p, o))$$

(by Def 3.4) $$= \{s, p, o\} \cup \varnothing$$

(by Def 3.3) $$= \{s, p, o\} \cup \varnothing$$

(by Def 3.5) $$= \{s, p, o\} \cup \varnothing$$

(by Def 3.1) $$= \{s, p, o\} \cup \varnothing$$

(by Def of \text{U}) $$= \{s, p, o\} \cup \varnothing$$

(by dist. of \text{U}) $$= \{s, p, o\} \cup \varnothing$$

(by Def 2.7) $$= \text{var}(\text{var}(s p o))$$

(by Def 2.7) $$= \text{var}(s p o)$$

$$\text{gp is gp}_1 \text{ gp}_2$$

$$\text{var}(\sigma(\text{gp}_1 \text{ gp}_2)) =$$

(by Def 3.5) $$= \text{var}(\sigma(\text{gp}_1) \cap \sigma(\text{gp}_2))$$

(by Def 3.7) $$= \text{var}(\sigma(\text{gp}_1) \cup \text{var}(\sigma(\text{gp}_2)))$$

(by induc.hyp.) $$= \text{var}(\text{gp}_1) \cup \text{var}(\text{gp}_2)$$

(by Def 3.1) $$= \text{var}(\text{gp}_1) \cup \text{var}(\text{gp}_2)$$

(by Def 2.7) $$= \text{var}(\text{gp}_1 \text{ gp}_2)$$

$$\text{gp is gp}_1 \text{ filter } R$$

$$\text{var}(\sigma(\text{gp}_1 \text{ filter } R)) =$$

(by Def 3.5) $$= \text{var}(\sigma(\text{gp}_1) \cap \sigma(R))$$

(by Def 3.7) $$= \text{var}(\sigma(\text{gp}_1) \cup \text{var}(\sigma(R)))$$

(by induc.hyp.) $$= \text{var}(\text{gp}_1) \cup \text{var}(\text{gp}1)$$

(by Def 3.1) $$= \text{var}(\text{gp}_1) \cup \text{var}(\text{gp}_2)$$

(by Def 2.7) $$= \text{var}(\text{gp}_1 \text{ filter } R)$$

$$\text{gp is gp}_1 \text{ diff } \text{gp}_2$$

$$\text{var}(\sigma(\text{gp}_1 \text{ diff } \text{gp}_2)) =$$

(by Def 3.5) $$= \text{var}(\sigma(\text{gp}_1) \cap \forall x \neg \sigma(gp_2))$$

(by Def 3.7) $$= \text{var}(\sigma(\text{gp}_1))$$

(by induc.hyp.) $$= \text{var}(\text{gp}_1)$$

(by Def 2.7) $$= \text{var}(\text{gp}_1 \text{ diff } \text{gp}_2)$$

$$\text{gp is gp}_1 \text{ minus } \text{gp}_2$$
If \( \text{var}(gp_1) \cap \text{var}(gp_2) = \emptyset \) then:

\[
\begin{align*}
\text{var}(\text{graph} \times \{gp_1\}) &= \text{var}(\text{graph} \times \{gp_1\}) \\
&= \bigcup_{i \in I_n} \{\text{var}(\sigma(r_{\text{graph}}((\sigma^{-1}(i))))(gp_1) \wedge \sigma(x) = i)\} \\
&= \bigcup_{i \in I_n} \{\text{var}(\sigma(gp_1)) \cup \{\sigma(x) = i\}\} \\
&= \text{var}(\text{graph} \times \{gp_1\}) \\
\end{align*}
\]

\( gp \) is graph \( \times \{gp_1\} \)

If \( I_n = \emptyset \), there is no active graph for matching \( gp_1 \). Then, by Definition 2.15, this case is equivalent to a case when the active graph is empty, which does not correspond to the assumption of this lemma. Therefore, we can assume that \( I_n \neq \emptyset \).

From \( I_n \neq \emptyset \), according to Definition 3.5, it holds that for all \( i \in I_n \) graph \( gr_o((\sigma^{-1}(i))) \) is a nonempty one.

\[
\begin{align*}
\text{var}(\text{graph} \times \{gp_1\}) &= \text{var}(\bigcup_{i \in I_n} \{\text{var}(\sigma(r_{\text{graph}}((\sigma^{-1}(i))))(gp_1) \wedge \sigma(x) = i)\}) \\
&= \text{var}(\bigcup_{i \in I_n} \{\text{var}(\sigma(gp_1)) \cup \{\sigma(x) = i\}\}) \\
&= \text{var}(\text{graph} \times \{gp_1\}) \\
\end{align*}
\]

\( \text{gp} \) is graph \( \times \{gp_1\} \)

If \( \sigma(i) \notin I_n \), by Definition 3.5, it holds \( i \notin \text{names}(0) \), i.e. by Definition 2.1. \( gr_o(i) = \emptyset \). Then, according to Definition 3.5, an active graph that is used for matching \( gp_1 \) is empty, which does not correspond to the assumption of this lemma. Therefore, we can assume that \( \sigma(i) \in I_n \).

From \( \sigma(i) \in I_n \), according to Definition 3.5, the graph \( gr_o(i) \) is a nonempty graph.

\[
\begin{align*}
\text{var}(\text{graph} \times \{gp_1\}) &= \text{var}(\bigcup_{i \in I_n} \{\text{var}(\sigma(r_{\text{graph}}((\sigma^{-1}(i))))(gp_1) \wedge \sigma(x) = i)\}) \\
&= \text{var}(\bigcup_{i \in I_n} \{\text{var}(\sigma(gp_1)) \cup \{\sigma(x) = i\}\}) \\
&= \text{var}(\text{graph} \times \{gp_1\}) \\
\end{align*}
\]

3.3. Modeling the Containment Relation

Modeling the query containment problem and subsumption problem include defining formulas \( \Theta \) and \( \Psi \), and relation \( \sim \), for containment, and relation \( \preceq \) for subsumption. These are defined in [56] and given in a short form in Figure 5.

Definition 3.8 specifies formula \( \Theta \), based on a query \( Q_1 \) (a candidate for sub-query), where \( \tilde{V}_1 \) denote variables from \( V \) that correspond to the relevant variables of \( Q_1 \), while \( \Phi_{2}(\tilde{V}_1) \) is a formula that corresponds to \( Q_1 \) according to Definition 3.6.

Definition 3.9 specifies formula \( \Psi \), based on queries \( Q_1 \) and \( Q_2 \), where \( \tilde{V}_2 \) denote variables from \( V \) that correspond to the relevant variables of \( Q_1 \), while \( \Phi_{1}(\tilde{V}_1) \) and \( \Phi_{2}(\tilde{V}_1) \) are formulas that correspond to \( Q_1 \) and \( Q_2 \), respectively according to Definition 3.6.

Relations \( \sim \) and \( \preceq \) are given within Definition 3.10. These relations connect queries \( Q_1 \) and \( Q_2 \) based on a relation between their relevant variables \( \tilde{V}_1 \) and \( \tilde{V}_2 \), respectively.

The following lemma connects the relevant variables of \( Q_1 \) and variables of \( Q_2 \) if relation \( \sim \) holds.
Lemma 3.6 Let $q_1$ and $q_2$ be queries, such that $q_2$ does not contain projections and $q_1 \sim q_2$ holds. Let $\overline{rv}_1$ be a set of relevant variables of $q_1$ and $\text{qpat}_2$ a query pattern of $q_2$. Then:

$$\overline{rv}_1 = \text{var}(\text{qpat}_2).$$

Proof. By Definition 3.10, from $q_1 \sim q_2$, it holds

$$\overline{rv}_1 = \overline{rv}_2.$$  

Then, by Lemma 2.2, it holds

$$\overline{rv}_1 = \text{var}(\text{qpat}_2) \cap \overline{dv}_2.$$  

By Definition 2.16, as $q_2$ does not contain projections, it holds

$$\overline{rv}_1 = \text{var}(\text{qpat}_2).$$ □

The following lemma is dual to Lemma 3.6, but considering relation $\sim$ instead of $\sim$ and without a restriction regarding projections of $q_2$.

Lemma 3.7 Let $q_1$ and $q_2$ be queries, such that $q_1 \sim q_2$. Let $\overline{rv}_1$ be a set of relevant variables of $q_1$ and $\text{qpat}_2$ and $\overline{dv}_2$ a query pattern and a set of distinguished variables of $q_2$, respectively. Then:

$$\overline{rv}_1 \subseteq \text{var}(\text{qpat}_2) \cap \overline{dv}_2.$$  

Proof. By Definition 3.10, from $q_1 \sim q_2$, it holds

$$\overline{rv}_1 \subseteq \overline{rv}_2.$$  

Then, by Lemma 2.2, it holds

$$\overline{rv}_1 \subseteq \text{var}(\text{qpat}_2) \cap \overline{dv}_2.$$ □

The following lemma simplifies the formula $\Psi$ in cases when the relation $\sim$ holds and $q_2$ does not contain projections.

Lemma 3.8 Let $q_1$ and $q_2$ be queries, let $\overline{rv}_1$ denote variables from $\Psi$ that correspond to the relevant variables of the query $q_1$, and let $\Phi_1(\overline{rv}_1)$ correspond to the query $q_1$. Let $\text{qpat}_2$ and $\overline{dv}_2$ be a query pattern and a query dataset of $q_2$, respectively. If $q_2$ does not contain projections and $q_1 \sim q_2$ holds, then formula $\Psi$ is equal to:

$$\forall \overline{rv}_1 \left( \Phi_1(\overline{rv}_1) \Rightarrow \sigma^{df(\text{qpat}_2)}(\text{qpat}_2) \right).$$

Proof. By Definition 3.9, $\Psi$ is defined as

$$\forall \overline{rv}_1 \left( \Phi_1(\overline{rv}_1) \Rightarrow \Phi_2(\overline{rv}_1) \right),$$

i.e. by Definition 3.6

$$\forall \overline{rv}_1 \left( \Phi_1(\overline{rv}_1) \Rightarrow \exists \overline{dv}_2 \sigma^{df(\text{qpat}_2)}(\text{qpat}_2) \right),$$

where $\overline{dv}_2$ is equal to $\text{var}(\sigma(\text{qpat}_2)) \setminus \overline{rv}_1$. By Definition 3.10, as $q_1 \sim q_2$, it holds $\overline{rv}_1 = \overline{rv}_2$, i.e. by Lemma 2.2,

$$\overline{rv}_1 = \text{var}(\text{qpat}_2) \cap \overline{dv}_2.$$  

As $q_2$ does not contain projections, it holds

$$\overline{rv}_1 = \text{var}(\text{qpat}_2),$$

i.e.

$$\text{var}(\text{qpat}_2) \setminus \overline{rv}_1 = \emptyset.$$  

Therefore, by Definition 3.1, it holds

$$\sigma(\text{var}(\text{qpat}_2)) \setminus \overline{rv}_1 = \emptyset,$$

i.e.

$$\sigma(\text{var}(\text{qpat}_2)) \subseteq \overline{rv}_1,$$

i.e. by Lemma 3.4,

$$\text{var}(\sigma(\text{qpat}_2)) \setminus \overline{rv}_1 = \emptyset.$$  

Therefore, $\overline{rv}_1$ is equal to an empty set. Then, the formula $\Psi$ is reduced to

$$\forall \overline{rv}_1 \left( \Phi_1(\overline{rv}_1) \Rightarrow \sigma^{df(\text{qpat}_2)}(\text{qpat}_2) \right).$$  

□

We propose the following reductions of the containment problem and the subsumption problem.

Procedure 3.1 (Logical formulation of the SPARQL query containment problem) Let $q_1$ and $q_2$ be queries, $\Theta$ be a formula generated from $q_1$ (Def 3.8) and $\Psi$ be a formula generated from $q_1$ and $q_2$ (Def 3.9). The query containment problem, i.e. if $q_1 \subseteq q_2$ holds, is reduced to a satisfiability of a disjunction of the following two conditions:

(1) $\Theta$ is valid

(2) $q_1 \sim q_2$ holds and $\Psi$ is valid.

Correctness of the procedure 3.1 is a direct consequence of proofs for theorems 4.11 (Section 4.2) and 4.13 (Section 4.3).

Procedure 3.2 (Logical formulation of the SPARQL query subsumption problem) Let $q_1$ and $q_2$ be queries, $\Theta$ be a formula generated from $q_1$ (Def 3.8) and $\Psi$ be a formula generated from $q_1$ and $q_2$ (Def 3.9). The query subsumption problem, i.e. if $q_1 \sqsubseteq q_2$ holds, is reduced to a satisfiability of a disjunction of the following two conditions:

(1) $\Theta$ is valid

(2) $q_1 \sim q_2$ holds and $\Psi$ is valid.

Correctness of the procedure 3.2 is a direct consequence of proofs for theorems 4.12 (Section 4.2) and 4.14 (Section 4.3).

4. Correctness of the Proposed Modeling

In this section, we prove the correctness of our modeling. We start with introducing necessary definitions and proving auxiliary lemmas (Section 4.1) and continue by proving soundness (Section 4.2) and completeness (Section 4.3) of the proposed reduction.
4.1. Interpretations and Models

There is a standard definition of $\mathcal{L}$-structure over a signature $\mathcal{L}$ given in model theory textbooks [11, 39]. The following definition represents its concretization over the SPARQL theory signature $\mathcal{L}$ corresponding to queries $Q_1$ and $Q_2$.

**Definition 4.1** ($\mathcal{L}$-structure $\mathfrak{D}$ corresponding to the queries $Q_1$, $Q_2$ and a dataset $\mathcal{D}$) An $\mathcal{L}$-structure corresponding to a dataset $\mathcal{D}$ over the SPARQL theory signature $\mathcal{L}$ corresponding to the queries $Q_1$ and $Q_2$, named $\mathfrak{D}$, is a pair $(D, I^D)$ where:

- $D$ is a nonempty domain equal to $\text{IF} \cup \{\sigma_i^{-1}(c) \mid c \in C\}$, where $I, B, L$ are sets of IRIs, blank nodes and literals respectively, appearing in the dataset $\mathcal{D}$,
- $I^D$ is a function that maps the non-logical terms of the signature in the following way:
  - The interpretation $I^D$ of constants from $C$ are from $\text{IF}$, and are defined as $I^D(c) \equiv (\sigma_i)^{-1}(c)$, for $c \in C$.
  - The interpretation $I^D$ of the function symbol datatype is a function $\text{DATATYPE} : \text{IF} \times 1 \rightarrow 1$, which is defined in the following way:
    
    $$\text{DATATYPE}(t) \equiv \begin{cases} \text{dt}(t), & t \in L \\ \text{err}, & \text{otherwise} \end{cases}$$

  - The interpretation $I^D$ of the predicate symbol isLiteral is a function $\text{ISLITERAL} : \text{IF} \rightarrow \text{bool}$, which is defined in the following way:
    
    $$\text{ISLITERAL}(t) = \text{T} \text{if and only if } t \in L$$

  - The interpretation $I^D$ of the predicate symbol isLiteral is a function $\text{B}_d : \text{IF} \times 1 \times \text{IF} \rightarrow \text{bool}$, which is defined in the following way:
    
    $$\text{B}_d(s, p, o) = \text{T} \text{if and only if } (s, p, o) \in d f(D)$$

  - The interpretation $I^D$ of the predicate symbol $\sigma_i$ is a function $\text{B}_\sigma : \text{IF} \times 1 \times \text{IF} \times 1 \rightarrow \text{bool}$, which is defined in the following way:
    
    $$\text{B}_\sigma(s, p, o, i) = \text{T} \text{if and only if } (s, p, o) \in g r_D(i)$$

Note that $\sigma_i$ is bijective by Definition 3.3, therefore $(\sigma_i)^{-1}$ is well defined. As a direct consequence of this definition, if $\text{B}_\sigma(s, p, o, i) = \text{T}$ then $i \in \text{names}(\mathcal{D})$. Otherwise, $\text{g r}_D(i)$ would be equal to $\sigma_0$ and $(s, p, o)$ does not belong to $\sigma_0$. Note also that the domain $D$ is at most countable.

A valuation of $\mathcal{L}$-formulas within $\mathcal{L}$-structure $\mathfrak{D}$, usually denoted by $v$, is a partial function from the set of variables $\mathcal{V}$ to $\text{IF}$.

For an $\mathcal{L}$-structure $\mathfrak{M}$, valuation $v$, and formula $\phi$ over the given SPARQL theory signature $\mathcal{L}$, we use standard notation $(\mathfrak{M}, v) \models Y$ denoting that $\mathfrak{M}$ with valuation $v$ is a model of formula $\phi$. If formula $\phi$ is a sentence, i.e. if it does not contain free variables, we use the common notation $\mathfrak{M} \models Y$. If a formula $Y$ contains variables $\bar{x}$, and a valuation $v$ maps them to $\bar{c}$, then $(\mathfrak{M}, v) \models Y(\bar{c})$ denotes that $I_v(Y(\bar{c}))$ is true.

There are standard definitions of embeddings, isomorphisms, and substructures given in model theory textbooks [11, 39]. The following definition represents their concretization over the presented signature $\mathcal{L}$ corresponding to $Q_1$ and $Q_2$.

**Definition 4.3** ($\mathcal{L}$-embedding, $\mathcal{L}$-isomorphism, $\mathcal{L}$-substructure) Let $\mathcal{M}$ and $\mathcal{N}$ be nonempty domains. Let $\mathfrak{M} = (M, I^M)$ and $\mathfrak{N} = (N, I^N)$ be $\mathcal{L}$-structures.

- **$\mathcal{L}$-embedding** An $\mathcal{L}$-embedding $\theta : \mathfrak{M} \rightarrow \mathfrak{N}$ is a function $\theta : M \rightarrow N$ with the following properties:
  
  (i) $\theta$ is injective,
  
  (ii) for relation symbols $\beta_d$ and $\beta_n$ in signature $\mathcal{L}$, and all $s, p, o, i \in M$,

  $$(s, p, o) \in I^M(\beta_d) \text{ if and only if } (\theta(s), \theta(p), \theta(o)) \in I^N(\beta_d),$$

  and

  $$(s, p, o, i) \in I^M(\beta_n) \text{ if and only if } (\theta(s), \theta(p), \theta(o), \theta(i)) \in I^N(\beta_n),$$

  (iii) for any constant symbol $c$ in the signature $\mathcal{L}$,

  $$\theta(I^M(c)) = I^N(c).$$

- **$\mathcal{L}$-isomorphism** An $\mathcal{L}$-isomorphism from $\mathfrak{M}$ to $\mathfrak{N}$ is a bijective $\mathcal{L}$-embedding from $\mathfrak{M}$ to $\mathfrak{N}$.

- **$\mathcal{L}$-substructure** $\mathfrak{N}$ is an $\mathcal{L}$-substructure of $\mathfrak{M}$ if $M \subseteq N$ and the inclusion map $\iota : M \rightarrow N$, such that $\iota(a) = a$ for all $a \in M$, is an $\mathcal{L}$-embedding.

Note that the definition of $\mathcal{L}$-embedding should provide the congruence of all relation symbols from $P_s$. The property (ii) should hold not only for $\beta_d$ and $\beta_n$, but also for the equality symbol, but it can be proven from property (i), i.e. from the injectiveness of $\theta$:

- for all $m_1, m_2 \in M$, if $m_1 = m_2$ then $\theta(m_1) = \theta(m_2)$, as $\theta$ is a function;
- the converse holds because $\theta$ is injective.

From the other side, if we include the property (ii) for the equality symbol, the property (i) is not necessary, as it is a corollary of the property (ii) for the equality symbol.

**Lemma 4.1** Let $\phi$ be a sentence over a given signature $\mathcal{L}$, and let $\mathfrak{D}'$ and $\mathfrak{D}''$ be $\mathcal{L}$-structures such that $\mathfrak{D}' \cong \mathfrak{D}''$. Then:

$$(\mathfrak{D}', \phi) \models Y \text{ if and only if } (\mathfrak{D}'', Y) \models Y$$

Proof. A proof of this lemma can be found in [39].

**Lemma 4.2** Let $\mathfrak{D}' = (D', I'^D)$ be an $\mathcal{L}$-structure over the SPARQL theory signature $\mathcal{L}$ that corresponds to $Q_1$ and $Q_2$. Let $\phi$ be a sentence over a given signature $\mathcal{L}$, such that $\mathfrak{D}' \models Y$. Then, there exists a dataset $\mathcal{D}$ and the corresponding $\mathcal{L}$-structure $\mathfrak{D} = (D, I^D)$ over the same signature, such that $\mathfrak{D} \models Y$. 

M. Spasić and M. Vujović Janičić: Soundness and Completeness of SPARQL Query Containment Solver SpeCS  
Page 13 of 32
Proof. According to Definition 4.3, $\mathcal{S}'$ has a nonempty domain $D'$, possibly infinite. By the downward Löwenheim-Skolem Theorem, there exists an $\mathcal{L}$-structure $\mathcal{S}'' = (D'', I^{D''})$ over the same signature, such that its domain $D''$ is countable and $\mathcal{S}'' \models \mathcal{Y}$.

Let us construct an RDF dataset $\mathcal{D}$ and the corresponding $\mathcal{L}$-structure $\mathcal{S} = (D, I^{D})$, such that it holds $\mathcal{S}'' \equiv \mathcal{S}$. First, we define a function $\theta : D'' \to \text{IBL}$ as follows:

- for all $c'' \in D''$ such that $c'' = I^{D''}(c)$ for some constant from $C$, $\theta(c'')$ is equal to $(\sigma_s)^{-1}(c)$. Note that the restriction of $\theta$ to the set of such $c''$ is injective, due to Definition 3.3.
- for all other $x$ from $D''$, $\theta(x)$ is equal to an arbitrary element from IBL, but taking into account that function $\theta$ should stay injective. This is possible, as $\mathcal{B}$ and $\mathcal{B}''$ are countable sets.

Then, we construct an RDF dataset $\mathcal{D}$ as follows:

1. For each constant $i$ from $D''$ we define graph $\mathcal{G}_i$, and function $g_{\mathcal{D}}$:
   
   $g_{\mathcal{D}}(\theta(i)) := c_i$
   
2. We define default graph $\mathcal{G}_d$:
   
   $g_{\mathcal{D}} := \{ (\theta(s), \theta(p), \theta(o)) \mid s, p, o \in D'' 
   \text{ and } I^{D''}(\beta)_D(s, p, o) \text{ is true} \}$

Then, dataset $\mathcal{D}$ contains default graph $\mathcal{G}_d$ and named graphs $\mathcal{G}_i$ with the corresponding names $\theta(i)$. Note that according to the modeling of queries where formula contains predicates $\beta_d$ and/or $\beta_p$ and the existence of its model in $\mathcal{S}''$, these graphs are not empty.

The function $\theta$ is an $\mathcal{L}$-embedding, according to Definition 4.3, because:

1. $\theta$ is injective, by its construction;
2. For all $s, p, o, i \in D''$, it holds $I^{D''}(\beta_d)(s, p, o) \iff (\theta(s), \theta(p), \theta(o)) \in g_{\mathcal{D}}$ (by constr.)
   (by Def. 4.1)
   (by Def. 4.1)
   (by Def. 4.1)
   (by Def. 4.1)
   and
3. $I^{D''}(\beta_p)(s, p, o, i) \iff (\theta(s), \theta(p), \theta(o)) \in g_{\mathcal{D}}(\theta(i))$
   (by constr.)
   (by Def. 4.1)
   (by Def. 4.1)
   (by Def. 4.1)

(iii) for each constant symbol $c$ of the signature $\mathcal{L}$, $I^{D''}(c) \equiv D''$, and $\theta(I^{D''}(c)) = I^{D}(c)$ by construction.

Function $\theta$ is surjective, because for each $x \in D$, there is an element $\exists x \in D''$ such that $\theta(x) = x$:

- if $x$ is the interpretation of a constant from $C$, according to Definition 4.1, construction of $\theta$ and the fact that $\mathcal{S}''$ and $\mathcal{S}$ are corresponding $\mathcal{L}$-structures, there is an element $x$ such that $\theta(x) = x$;
- otherwise, each other element from $D$ is an image of an element from $D''$ by the construction of $\theta$ and the dataset $D$.

Therefore, $\mathcal{L}$-embedding $\theta$ is bijective, and by Definition 4.3, it holds $\mathcal{S}'' \equiv \mathcal{S}$. Then, by Lemma 4.1, it holds $\mathcal{D} \models \mathcal{Y}$.

Definition 4.4 (Relations $\rightarrow, \leftarrow, \leftrightarrow$) Let $\mu : \mathcal{B} \to \text{IBL}$ be a mapping and $\nu : \mathcal{Y} \to \text{IBL}$ be a valuation.

1. Mapping $\mu$ defines a valuation $\nu$, denoted $\mu \rightarrow \nu$, if $\sigma_t(\text{dom}(\mu)) \subseteq \text{dom}(\nu)$ and for each $x \in \text{dom}(\mu)$ it holds $\nu(\sigma_t(x)) = \mu(x)$.
2. Mapping $\mu$ is defined by a valuation $\nu$, denoted $\mu \leftarrow \nu$, if $\text{dom}(\mu) \supseteq (\sigma_t)^{-1}(\text{dom}(\nu))$ and for each $x \in \text{dom}(\nu)$ it holds $\mu((\sigma_t)^{-1}(x)) = \nu(x)$.
3. Mapping $\mu$ corresponds to valuation $\nu$, denoted $\mu \leftrightarrow \nu$, if $\mu \rightarrow \nu$ and $\mu \leftarrow \nu$.

These relations are well defined as $\sigma_t$ is bijective. Note that if $\mu \rightarrow \nu$ holds, $\text{dom}(\mu) \supseteq (\sigma_t)^{-1}(\text{dom}(\nu))$ and $\text{dom}(\nu) = \text{dom}(\mu)$. Similarly to the notation $\mu_{x \leftarrow c}$, we introduce $\nu_{x \rightarrow c}$ denoting a valuation such that $\text{dom}(\nu_{x \rightarrow c}) \equiv \{ x \}$ and $\nu_{x \rightarrow c}(x) \equiv \sigma_t(x)$.

Note that, by Definition 4.4, $\mu_{x \rightarrow c} \leftrightarrow \nu_{x \rightarrow c}$, as $\text{dom}(\nu) = \{ x \} = \{ \sigma_t(x) \} = \sigma_t(\text{dom}(\mu))$ and $\nu(x) = \sigma_t(x) = \mu(x)$.

Definition 2.8 can be used in the context of valuations as well: $\nu_1$ and $\nu_2$ are compatible, denoted by $\nu_1 \simeq \nu_2$, if for each $x$ such that $x \in \text{dom}(\nu_1) \cap \text{dom}(\nu_2)$ it holds that $\nu_1(x) = \nu_2(x)$.

Lemma 4.3 Let $\mu_1, \mu_2 : \mathcal{B} \to \text{IBL}$ be mappings, and let $\nu_1, \nu_2 : \mathcal{Y} \to \text{IBL}$ be valuations such that $\mu_1 \rightarrow \nu_1$ and $\mu_2 \rightarrow \nu_2$. Then:

$$\mu_1 \simeq \mu_2$$

if and only if $\nu_1 \simeq \nu_2$.

Proof. ($\Rightarrow$) If $\text{dom}(\nu_1) \cap \text{dom}(\nu_2)$ is an empty set, by Definition 2.8, it holds $\nu_1 \simeq \nu_2$. Otherwise, let $x$ be an element of $\text{dom}(\nu_1) \cap \text{dom}(\nu_2)$, i.e. $x \in \text{dom}(\nu_1)$ and $x \in \text{dom}(\nu_2)$. From $\mu_1 \leftrightarrow \nu_1$ and $\mu_2 \leftrightarrow \nu_2$, by Definition 4.4, it holds $x \in \{ \sigma_t(x) \mid x \in \text{dom}(\mu_1) \}$ and $x \in \{ \sigma_t(x) \mid x \in \text{dom}(\mu_2) \}$. Therefore, since $\sigma_t$ is a bijective function, there exists a unique $x$, such that $x = \sigma_t(x)$, $x \in \text{dom}(\mu_1)$ and $x \in \text{dom}(\mu_2)$, i.e. $x \in \text{dom}(\mu_1) \cap \text{dom}(\mu_2)$. As it holds $\mu_1 \simeq \mu_2$ (Definition 2.8), it holds $\mu_1(x) = \mu_2(x)$. 

M. Spasić and M. Vujošević Janičić: Soundness and Completeness of SPARQL Query Containment Solver SpeCS

Page 14 of 32
and by the assumptions $\mu_1 \leftrightarrow v_1$ and $\mu_2 \leftrightarrow v_2$, it holds
\[ v_1(\sigma_i(x)) = \mu_1(x) \text{ and } \mu_2(x) = v_2(\sigma_i(x)). \]

Therefore, it holds
\[ v_1(\sigma_i(x)) = v_2(\sigma_i(x)), \text{ i.e. } v_1(x) = v_2(x). \]

By Definition 2.8, it holds $v_1 \simeq v_2$.

(⇐) If $\text{dom}(\mu_1) \cap \text{dom}(\mu_2)$ is an empty set, by Definition 2.8, it holds $\mu_1 \simeq \mu_2$. Otherwise, let $x$ be an element of $\text{dom}(\mu_1) \cap \text{dom}(\mu_2)$, i.e. $x \in \text{dom}(\mu_1)$ and $x \in \text{dom}(\mu_2)$.

From $\mu_1 \leftrightarrow v_1$ and $\mu_2 \leftrightarrow v_2$, by Definition 4.4, it holds $x \in \{\sigma_i^{-1}(x) \mid x \in \text{dom}(v_1)\}$ and $x \in \{\sigma_i^{-1}(x) \mid x \in \text{dom}(v_2)\}$. Therefore, since $(\sigma_i^{-1})$ is a bijective function, there exists a unique $x$, such that
\[ x = (\sigma_i^{-1}(x), x \in \text{dom}(v_1) \text{ and } x \in \text{dom}(v_2) \]
i.e. $x \in \text{dom}(v_1) \cap \text{dom}(v_2)$. As it holds $v_1 \simeq v_2$ (Definition 2.8), it holds
\[ v_1(x) = v_2(x) \]
and by the assumptions $\mu_1 \leftrightarrow v_1$ and $\mu_2 \leftrightarrow v_2$, it holds
\[ \mu_1((\sigma_i^{-1}(x)) = v_1(x) \text{ and } v_2(x) = \mu_2((\sigma_i^{-1}(x)). \]

Therefore, it holds
\[ \mu_1((\sigma_i^{-1}(x)) = \mu_2((\sigma_i^{-1}(x)), \text{ i.e. } \mu_1(x) = \mu_2(x). \]

By Definition 2.8, it holds $\mu_1 \simeq \mu_2$. \hfill \Box

**Lemma 4.4** Let $\mu, \mu_1, \mu_2 : \mathcal{V} \rightarrow \mathcal{IBL}$ be mappings, and
$\nu, \nu_1, \nu_2 : \mathcal{V} \rightarrow \mathcal{IBL}$ be valuations such that $\mu \leftrightarrow \nu$, $\mu_1 \leftrightarrow \nu_1$ and $\mu_2 \leftrightarrow \nu_2$. Then:
\[ \begin{aligned}
\mu_1 & \simeq \mu_2 \text{ and } \mu = \mu_1 \cup \mu_2 \\
& \text{if and only if} \\
v_1 & \simeq v_2 \text{ and } \nu = v_1 \cup v_2.
\end{aligned} \]

**Proof.** (⇒) From $\mu_1 \leftrightarrow \nu_1, \mu_2 \leftrightarrow \nu_2$ and $\mu_1 \simeq \mu_2$, by Lemma 4.3 it holds that
\[ v_1 \simeq v_2. \]

Therefore, $\nu_1 \cup v_2$ is a well defined valuation.

From $\mu \leftrightarrow \nu, \mu_1 \leftrightarrow \nu_1$ and $\mu \simeq \mu_1$ (as $\mu$ is an extension of $\mu_1$), by Lemma 4.3 it holds that
\[ \nu \simeq v_1. \]

Similarly, from $\mu \leftrightarrow \nu, \mu_2 \leftrightarrow \nu_1$, and $\mu \simeq \mu_2$ (as $\mu$ is an extension of $\mu_2$), by Lemma 4.3, it holds that
\[ \nu \simeq v_2. \]

Also,
\[ \begin{aligned}
\text{dom}(\nu) & = \\
& \text{(by Def 4.4 as } \mu \leftrightarrow \nu) = (\sigma_{\nu})^{-1}(\text{dom}(\nu)) \\
& \text{(as } \nu = v_1 \cup v_2) = (\sigma_1)^{-1}(\text{dom}(v_1) \cup \text{dom}(v_2)) \\
& \text{(by Def 3.1)} = (\sigma_1)^{-1}(\text{dom}(v_1)) \cup (\sigma_2)^{-1}(\text{dom}(v_2)) \\
& \text{(by Def 4.4, as } \mu_1 \leftrightarrow \nu_1 \text{ and } \mu_2 \leftrightarrow \nu_2) = \text{dom}(\mu_1) \cup \text{dom}(\mu_2) \\
& \text{(} \mu_1 \text{ and } \mu_2 \text{ are comp.}) = \text{dom}(\mu_1 \cup \mu_2) \\
& \text{As } \text{dom}(\mu) = \text{dom}(\mu_1) \cup \text{dom}(\mu_2), \text{ for } x \in \text{dom}(\mu), \text{ it holds} \\
x \in \text{dom}(\mu_1) \text{ or } x \in \text{dom}(\mu_2). \\
& \text{From } \mu \simeq \mu_1 \text{ and } \mu \simeq \mu_2, \text{ by Definition 2.8, it holds} \\
\mu(x) = \mu_1(x) \text{ or } \mu(x) = \mu_2(x). \\
& \text{Therefore, as } \mu_1 \simeq \mu_2, \\
x \in \text{dom}(\mu) \text{ implies } \mu(x) = (\mu_1 \cup \mu_2)(x). \\
& \text{Finally, from } \text{dom}(\mu) = \text{dom}(\mu_1 \cup \mu_2) \text{ and the last equation, it holds} \\
\mu = \mu_1 \cup \mu_2. \]

\hfill \Box
**Definition 4.5** (Notation $[[{-}]]_\mu$.) Let $v$ be a valuation, and $t$ a term over SPARQL theory signature $\mathcal{L}$. A value of the term $t$, according to the valuation $\nu$, in notation $[[t]]_\nu$, is a value from IBL, defined in the following way:

$$[[t]]_\nu \stackrel{\text{def}}{=} \begin{cases} \text{err}, & t \in \mathcal{V} \text{ and } t \notin \text{dom}(\nu) \\ \text{err}, & t \text{ is } \text{datatype}(t_1) \text{ and } [[t_1]]_\nu = \text{err} \\ \text{err}, & t \text{ is } \text{isLiteral}(t_1) \text{ and } [[t_1]]_\nu = \text{err} \\ I_1(t), & \text{otherwise.} \end{cases}$$

**Lemma 4.5** Let $E$ be an expression. Let $\mu : \mathcal{V} \rightarrow \mathcal{BL}$ be a mapping and $v : \mathcal{V} \rightarrow \mathcal{IBL}$ be a valuation such that $\mu \equiv v$. Then:

$$[[E]]_\mu = [[\sigma(E)]]_\nu.$$  

**Proof.** The lemma is proved by induction on an expression $E$.

- $E$ is $c$, $c \in \mathcal{IL}$
  $$[[c]]_\mu = \begin{cases} (\text{by Def } 2.13) & = c \\ (\text{by Def } 3.3) & = (\sigma_1)^{-1}(\sigma_2(c)) \\ (\text{by Def } 4.1) & = I^D(\sigma(c)) \\ (\text{by Def } 3.5) & = [[\sigma(c)]]_\nu \end{cases}$$

- $E$ is $v$, $v \in \mathcal{V}$
  $$[[v]]_\mu = \begin{cases} (\text{by Def } 2.13, \text{as } v \in \text{dom}(\mu)) & = \mu(v) \\ (\text{by Def } 4.4 \text{ as } \mu \equiv v) & = v(\sigma_1(v)) \\ (\text{by Def } 3.5) & = v(\sigma(v)) \\ (\text{by Def } 4.2) & = I_1(\sigma(v)) \\ (\text{by Def } 4.5, \text{as } \sigma(v) \in \text{dom}(\nu)) & = [[\sigma(v)]]_\nu \end{cases}$$

- $E$ is datatype($E_1$)
  $$[[\text{datatype}(E_1)]]_\mu = \begin{cases} (\text{by Def } 2.13, \text{as } v \notin \text{dom}(\mu)) & = \text{err} \\ (\text{by Def } 4.5, \text{as } \sigma(v) \notin \text{dom}(\mu)) & = [[\sigma(v)]]_\nu \\ \text{by induction hypothesis, it holds } [[E_1]]_\mu = \text{err} \iff [[\sigma(E_1)]]_\nu = \text{err}.

\[\square\]

**Lemma 4.6** Let $E$ be an expression, and $v$ a variable. Let $\mu : \mathcal{V} \rightarrow \mathcal{BL}$ be a mapping and $\nu : \mathcal{V} \rightarrow \mathcal{IBL}$ be a valuation such that $\mu \equiv v$. Then:

$$\mu(v) = [[E]]_\mu \text{ if and only if } (\mathcal{D}, v) \vdash \sigma(v) = \sigma(E).$$

**Proof.**

- $\mu(v) = [[E]]_\mu \iff$ (by Lemma 4.5)
  $$\begin{cases} (\text{by Def } 4.5) & \mu(v) = [[\sigma(E)]]_\nu \\ (\text{by Def } 4.4 \text{ as } \mu \equiv v) & \mu(v) = I_1(\sigma(E)) \\ (\text{by Def } 4.2) & \mu(v) = \nu(\sigma(E)) \end{cases}$$

\[\square\]

**Lemma 4.7** Let $\mu : \mathcal{V} \rightarrow \mathcal{BL}$ be a mapping and $\nu : \mathcal{V} \rightarrow \mathcal{IBL}$ be a valuation such that $\mu \equiv v$. Then:

$$\mu \vdash R \iff (\mathcal{D}, v) \vdash \sigma(R).$$

**Proof.** The lemma is proved by induction on a condition $R$.

- $R$ is $E_1 = E_2$
  $$\mu \vdash E_1 = E_2 \iff (\text{by Def } 2.14) \mu \vdash [[E_1]]_\mu \neq \text{err} \text{ and }$$
**Proof.** By Lemma 2.1, from $\mu \in \mathbb{L}_g$, it holds $\text{dom}(\mu) = \text{var}(\mu)$. Therefore, by Definitions 3.5 and 4.4, as $\mu \leftrightarrow \nu$, it holds $\text{dom}(\nu) = \sigma(\text{dom}(\mu)) = \sigma(\text{var}(\mu))$. Then, by Lemma 3.4, it holds $\text{dom}(\nu) = \sigma(\text{var}(\mu))$. The rest of specified equivalence is proved by induction over graph pattern $\mu$.

$\mathbb{G}$ is $\mathbb{T}$ where $\mathbb{T}$ is a query dataset, $\mathbb{G}$ a graph within $\mathbb{D}$, and $\mu$ a graph pattern. Let $\mu : \mathbb{V} \rightarrow \mathbb{I}B\mathbb{L}_1$ be a mapping and $v : \mathbb{V} \rightarrow \mathbb{I}B\mathbb{L}_1$ be a valuation such that $\mu \leftrightarrow \nu$. Then:

$$\mu \in \mathbb{L}_g$$

if and only if

$$(\mathfrak{D}, \nu) \models \sigma(\mu) \text{ and } \text{dom}(\nu) = \text{var}(\sigma(\mu)).$$

**Lemma 4.8** Let $\mathbb{D}$ be a query dataset, $\mathbb{G}$ a graph within $\mathbb{D}$, and $\mu$ a graph pattern. Let $\mu : \mathbb{V} \rightarrow \mathbb{I}B\mathbb{L}_1$ be a mapping and $v : \mathbb{V} \rightarrow \mathbb{I}B\mathbb{L}_1$ be a valuation such that $\mu \leftrightarrow \nu$. Then:

$$\mu \in \mathbb{L}_g$$

if and only if

$$(\mathfrak{D}, \nu) \models \sigma(\mu) \text{ and } \text{dom}(\nu) = \text{var}(\sigma(\mu)).$$

**Proof.** By Lemma 2.1, from $\mu \in \mathbb{L}_g$, it holds $\text{dom}(\mu) = \text{var}(\mu)$. Therefore, by Definitions 3.5 and 4.4, as $\mu \leftrightarrow \nu$, it holds $\text{dom}(\nu) = \sigma(\text{dom}(\mu)) = \sigma(\text{var}(\mu))$. Then, by Lemma 3.4, it holds $\text{dom}(\nu) = \sigma(\text{var}(\mu))$. The rest of specified equivalence is proved by induction over graph pattern $\mu$.

$\mathbb{G}$ is $\mathbb{T}$ where $\mathbb{T}$ is a query dataset, $\mathbb{G}$ a graph within $\mathbb{D}$, and $\mu$ a graph pattern. Let $\mu : \mathbb{V} \rightarrow \mathbb{I}B\mathbb{L}_1$ be a mapping and $v : \mathbb{V} \rightarrow \mathbb{I}B\mathbb{L}_1$ be a valuation such that $\mu \leftrightarrow \nu$. Then:

$$\mu \in \mathbb{L}_g$$

if and only if

$$(\mathfrak{D}, \nu) \models \sigma(\mu) \text{ and } \text{dom}(\nu) = \text{var}(\sigma(\mu)).$$

**Proof.** By Lemma 2.1, from $\mu \in \mathbb{L}_g$, it holds $\text{dom}(\mu) = \text{var}(\mu)$. Therefore, by Definitions 3.5 and 4.4, as $\mu \leftrightarrow \nu$, it holds $\text{dom}(\nu) = \sigma(\text{dom}(\mu)) = \sigma(\text{var}(\mu))$. Then, by Lemma 3.4, it holds $\text{dom}(\nu) = \sigma(\text{var}(\mu))$. The rest of specified equivalence is proved by induction over graph pattern $\mu$.

$\mathbb{G}$ is $\mathbb{T}$ where $\mathbb{T}$ is a query dataset, $\mathbb{G}$ a graph within $\mathbb{D}$, and $\mu$ a graph pattern. Let $\mu : \mathbb{V} \rightarrow \mathbb{I}B\mathbb{L}_1$ be a mapping and $v : \mathbb{V} \rightarrow \mathbb{I}B\mathbb{L}_1$ be a valuation such that $\mu \leftrightarrow \nu$. Then:

$$\mu \in \mathbb{L}_g$$

if and only if

$$(\mathfrak{D}, \nu) \models \sigma(\mu) \text{ and } \text{dom}(\nu) = \text{var}(\sigma(\mu)).$$
By Definitions 4.1, 4.2 and 4.5, it holds

$$(\mathfrak{D}, v) \vDash \beta_n(\sigma(s), \sigma(p), \sigma(o), i),$$
i.e.

$$\mathfrak{D}, v \vDash \bigvee_{i_j \in cx(\mathfrak{D})} \beta_n(\sigma(s), \sigma(p), \sigma(o), i_j).$$

Putting all options together, by Definition 3.5, it holds

$$(\mathfrak{D}, v) \vDash \sigma^\mathfrak{D}(s \circ p \circ o).$$

$$\iff$$ By Lemma 3.4, from $\text{dom}(v) = \text{var}(\sigma^\mathfrak{D}(tp))$, it holds $\text{dom}(v) = \sigma(\text{var}(tp))$. By Definitions 4.4, as $\mu \mapsto v$, it holds

$$\text{dom}(\mu) = (\sigma^{-1}(\text{dom}(v))) = (\sigma^{-1}(\sigma(\text{var}(tp)))) = \text{var}(tp)$$

From $\mathfrak{D}, v \vDash \sigma^\mathfrak{D}(s \circ p \circ o)$, by Definition 3.5, one option of the following two holds:

$$(\mathfrak{D}, v) \vDash \beta_d(\sigma(s), \sigma(p), \sigma(o), i).$$

if $g \not\in \text{dom}(cx)$ (4)

$$(\mathfrak{D}, v) \vDash \bigvee_{i_j \in cx(\mathfrak{D})} \beta_d(\sigma(s), \sigma(p), \sigma(o), i_j).$$

if $g \in \text{dom}(cx)$ (5)

From option (4), by Definitions 4.1, 4.2 and 4.5, it holds

$$B_d([[\sigma(s)]_\mu], [[\sigma(p)]_\mu], [[\sigma(o)]_\mu]) = \top.$$ 

Therefore, by Lemma 4.5, from $\mu \mapsto v$, it holds

$$B_d([[s]]_\mu, [[p]]_\mu, [[o]]_\mu) = \top.$$ 

Then, by Definition 4.1, it holds

$$[[s]]_\mu [[p]]_\mu [[o]]_\mu \in df(\mathfrak{D}),$$
i.e. as $g \not\in \text{dom}(cx)$,

$$[[s]]_\mu [[p]]_\mu [[o]]_\mu \in \mathfrak{g}.$$ 

From option (5), there exists $i_j \in cx(g)$, $\text{gr}_d(i_j) = \mathfrak{g}_j$, such that

$$(\mathfrak{D}, v) \vDash \beta_n(\sigma(s), \sigma(p), \sigma(o), i_j).$$

By Definitions 4.1, 4.2 and 4.5, it holds:

$$B_n([[\sigma(s)]_\mu], [[\sigma(p)]_\mu], [[\sigma(o)]_\mu], i_j) = \top.$$ 

Therefore, by Lemma 4.5, from $\mu \mapsto v$, it holds

$$B_n([[s]]_\mu, [[p]]_\mu, [[o]]_\mu, i_j) = \top.$$ 

By Definition 4.1, it holds

$$[[s]]_\mu [[p]]_\mu [[o]]_\mu \in \text{gr}_d((\sigma^{-1}(i_j)).$$

By Definition 2.13, it holds $[[s \circ p \circ o]]_\mu \in \mathfrak{g}_j$ for some $\mathfrak{g}_j$ that forms $\mathfrak{g}$, i.e. $[[s \circ p \circ o]]_\mu \in \mathfrak{g}$. Therefore, putting all options together, by Definition 2.15, as $\text{dom}(\mu) = \text{var}(tp)$, it holds

$$\mu \in [[s \circ p \circ o]]_\mathfrak{g}.$$ 

$\iff$ By Definition 2.9, $\mu = \mu_1 \cup \mu_2$, where $\mu_1 \in [[sp1]]_\mathfrak{g}$, $\mu_2 \in [[sp2]]_\mathfrak{g}$ and $\mu_1 \approx \mu_2$. By induction hypotheses on $\text{gp}_1$ and $\text{gp}_2$, mappings $\mu_1$ and $\mu_2$, and their corresponding valuations $v_1$ and $v_2$, where $\mu_1 \mapsto v_1$ and $\mu_2 \mapsto v_2$, it holds.\footnote{Until the end of this proof, we use notation $\sigma$ instead of $\sigma^\mathfrak{D}$ for brevity.}

$$(\mathfrak{D}, v_1) \vDash \sigma(\text{gp}_1) \text{ and } (\mathfrak{D}, v_2) \vDash \sigma(\text{gp}_2).$$

By Lemma 4.4, from $\mu_1 \approx \mu_2$, $\mu = \mu_1 \cup \mu_2$, $\mu_1 \mapsto v_1$, $\mu_2 \mapsto v_2$ and $\mu \mapsto v$, it holds $v_1 \approx v_2$ and $v = v_1 \cup v_2$. Therefore, $v \approx v_1$ and $v \approx v_2$, and it holds

$$(\mathfrak{D}, v) \vDash \sigma(\text{gp}_1) \text{ and } (\mathfrak{D}, v) \vDash \sigma(\text{gp}_2),$$
i.e. by Definition 4.2,

$$(\mathfrak{D}, v) \vDash \sigma(\text{gp}_1) \land \sigma(\text{gp}_2).$$

Then, by Definition 3.5, it holds

$$(\mathfrak{D}, v) \vDash \sigma(\text{gp}_1, \text{gp}_2).$$

$$\iff$$ By Definition 3.5, from $\mathfrak{D}, v \vDash \sigma(\text{gp}_1, \text{gp}_2)$, it holds

$$(\mathfrak{D}, v) \vDash \sigma(\text{gp}_1) \land \sigma(\text{gp}_2),$$
i.e. by Definition 4.2,

$$(\mathfrak{D}, v) \vDash \sigma(\text{gp}_1) \text{ and } (\mathfrak{D}, v) \vDash \sigma(\text{gp}_2).$$

All free variables appearing in these two formulas belong to $\text{var}(\sigma(\text{gp}_1))$ and $\text{var}(\sigma(\text{gp}_2))$, while restrictions of $\nu$ to these domains are denoted $v_1$ and $v_2$, respectively. Note that it holds

$$\text{dom}(v) = \text{dom}(v_1) \cup \text{dom}(v_2) \text{ and } v = v_1 \cup v_2.$$ 

Therefore, models $\mathfrak{D}, v_1$ and $\mathfrak{D}, v_2$ can be used in the previous formulas:

$$(\mathfrak{D}, v_1) \vDash \sigma(\text{gp}_1) \text{ and } (\mathfrak{D}, v_2) \vDash \sigma(\text{gp}_2).$$

By induction hypotheses on $\text{gp}_1$ and $\text{gp}_2$, valuations $v_1$ and $v_2$, and their corresponding mappings $\mu_1$ and $\mu_2$, such that $\mu_1 \mapsto v_1$ and $\mu_1 \mapsto v_1$, as $\text{dom}(v_1) = $$
$\var{\sigma(g_1)}$ and $\text{dom}(v_2) = \var{\sigma(g_2)}$, it holds $\mu_1 \in \llbracket g_1 \rrbracket_\sigma^D$ and $\mu_2 \in \llbracket g_2 \rrbracket_\sigma^D$. By Lemma 4.4, from $v_1 \simeq v_2$, $v = v_1 \cup v_2$, $\mu \leftrightarrow v_1$, $\mu_1 \leftrightarrow v_1$ and $\mu_2 \leftrightarrow v_2$, it holds $\mu_1 \simeq \mu_2$ and $\mu = \mu_1 \cup \mu_2$. Therefore, by Definition 2.9 of $\llbracket \cdot \rrbracket_\sigma$, it holds

$$\mu \in \llbracket g_1 \rrbracket_\sigma^D \mathrel{\bowtie} \llbracket g_2 \rrbracket_\sigma^D.$$  

Then, by Definition 2.15, it holds

$$\mu \in \llbracket g_{1, \cdot, g_2} \rrbracket_\sigma^D.$$  

gp is $g_1$ filter $R$  

$$\mu \in \llbracket g_1 \rrbracket_\sigma^D$$  

(by Def 2.15) iff $\mu \in \llbracket g_1 \rrbracket_\sigma^D$ and $\mu \vdash R$

(by induc.hyp.) iff $\langle \sigma, \nu \rangle \vdash \sigma(g_1)$ and dom$(\nu) = \var{\sigma(g_1)}$

(by Lemma 4.7) iff $\langle \sigma, \nu \rangle \vdash \sigma(g_1)$ and dom$(\nu) = \var{\sigma(g_1)}$ $(\langle \sigma, \nu \rangle \vdash \sigma(R))$

(by Def 4.2) iff $\langle \sigma, \nu \rangle \vdash \sigma(g_1) \land \sigma(R)$ and dom$(\nu) = \var{\sigma(g_1)}$

(by Def 4.2, as $\langle \sigma, \nu \rangle \vdash \sigma(g_1) \land \sigma(R)$ and var$(\sigma) \subseteq \var{\sigma(g_1)}$) dom$(\nu) = \var{\sigma(g_1)}$

(by Def 3.5) iff $\langle \sigma, \nu \rangle \vdash \sigma(g_1) \land \sigma(R)$ dom$(\nu) = \var{\sigma(g_1)}$

Let us prove by contraposition that it holds $\langle \sigma, \nu \rangle \vdash \forall \nu \neg \var{\sigma(g_2)}$, i.e. let us assume

$$\langle \sigma, \nu \rangle \vdash \exists \nu \var{\sigma(g_2)}.$$  

This means that we can extend the valuation $\nu$ to all the variables from $\var{\sigma(g_2)}$. (by adding values to the variables from $\mathcal{X}$, and then restrict it only to the variables from $\var{\sigma(g_2)}$. (by removing variables from $\var{\sigma(g_1)}$ that are not in $\var{\sigma(g_2)}$). Let $v_2$ denote such valuation. Then, it holds

$$\langle \sigma, \nu \rangle \vdash \forall \nu \neg \var{\sigma(g_2)}.$$  

Note that it holds $v_2 \simeq v$ by construction. From the induction hypothesis applied on $v_2$ and a mapping $\mu_2$ such that $\mu_2 \leftrightarrow v_2$, it holds $\mu_2 \in \llbracket g_2 \rrbracket_\sigma^D$. By Lemma 4.3, from $\mu \leftrightarrow v$, $\mu_2 \leftrightarrow v_2$ and $v \simeq v_2$, it holds $\mu \simeq \mu_2$. This is a contradiction with nonexistence of a mapping compatible with $\mu$ and from $\llbracket g_2 \rrbracket_\sigma^D$. Therefore, it holds:

$$\langle \sigma, \nu \rangle \vdash \forall \nu \neg \var{\sigma(g_2)}.$$  

and then by Definition 4.2

$$\langle \sigma, \nu \rangle \vdash \sigma(g_1) \land \forall \nu \neg \var{\sigma(g_2)}.$$  

By Definition 3.5, it holds

$$\langle \sigma, \nu \rangle \vdash \sigma(g_1) \land \forall \nu \neg \var{\sigma(g_2)}.$$  

By Lemma 3.4, from dom$(\nu) = \var{\sigma(g_1)}$, it holds dom$(\nu) = \var{\sigma(g_1)}$, and then by Definition 2.7 it holds dom$(\nu) = \var{\sigma(g_1) \text{ diff } g_2}$. Therefore, by Lemma 3.4, it holds dom$(\nu) = \var{\sigma(g_1) \text{ diff } g_2})$.

(\leftrightarrow) By Definition 3.5, from

$$\var{\sigma(g_1) \text{ diff } g_2}$$  

it holds

$$\var{\sigma(g_1) \land \forall \nu \neg \var{\sigma(g_2)}}.$$  

i.e. by Definition 4.2,

$$\langle \sigma, \nu \rangle \vdash \sigma(g_1) \land \forall \nu \neg \var{\sigma(g_2)}.$$  

By Definition 3.7, as $\mathcal{X}$ denotes all variables that appear in $\sigma(g_2)$, but not in $\sigma(g_1)$, it holds dom$(\nu) = \var{\sigma(g_1)}$. Applying the induction hypothesis on the valuation $\nu$ and its corresponding mapping $\mu (\mu \leftrightarrow v)$, it holds $\mu \in \llbracket g_2 \rrbracket_\sigma^D$.

It should also be proved that there does not exist a $\mu_2$ such that $\mu_2 \in \llbracket g_2 \rrbracket_\sigma^D$ and $\mu_2 \simeq \mu$. This can be proved by contradiction, i.e. let us assume the existence of such a mapping $\mu_2$. By induction hypothesis, applied
on the mapping $\mu_2$ and a valuation $v_2$ such that $\mu_2 \leftrightarrow v_2$, it holds

$$\langle \varPi, v_2 \rangle \models \sigma(gp_2).$$

By Lemma 4.3, from $\mu_2 \approx \mu$, $\mu \leftrightarrow v$ and $\mu_2 \leftrightarrow v_2$, it holds $v_2 \approx v$. Then, $v \cup v_2$ is a well defined valuation, denoted by $v'$. As $v'$ is an extension of $v_2$, from $(\varPi, v_2) \models \sigma(gp_2)$, it holds

$$\langle \varPi, v' \rangle \models \sigma(gp_2).$$

As $v'$ is an extension of $v$, from $(\varPi, v) \models \forall \varphi \neg \sigma(gp_2)$, it holds

$$(\varPi, v') \models \forall \varphi \neg \sigma(gp_2).$$

Therefore, there is an extension $v''$ of $v'$, such that

$$\langle \varPi, v'' \rangle \models \neg \sigma(gp_2).$$

Also, as $v''$ is an extension of $v'$, from $(\varPi, v') \models \sigma(gp_2)$, it holds

$$\langle \varPi, v'' \rangle \models \sigma(gp_2).$$

Therefore, $(\varPi, v'')$ is a model of formula $\sigma(gp_2)$ and its negation as well, which is a contradiction.

As it holds $\mu \in [gp_1]_c$ and it holds that there does not exist $\mu_2$ such that $\mu_2 \in [gp_2]_c$ and $\mu_2 \approx \mu$, then, by Definition 2.9

$$\mu \in [gp_1]_c \setminus [gp_2]_c.$$  

Then, by Definition 2.15, it holds

$$\mu \notin [gp_1]_c \setminus [gp_2]_c.$$  

Then, by Definition 2.15, it holds

$$\mu \notin \{gp_1 \text{ minus } gp_2\}_c.$$  

gp is gp1 minus gp2

If $\text{var}(gp_1) \cap \text{var}(gp_2) = \emptyset$ then:

$(\Rightarrow)$ By Definition 2.15, from

$$\mu \in [gp_1]_c \setminus [gp_2]_c,$$

it holds $\mu \notin [gp_1]_c \setminus [gp_2]_c$. By Definition 2.9, it holds $\mu \notin [gp_1]_c$. By induction hypothesis, from $\mu \notin [gp_1]_c$, for mapping $\mu$ and a corresponding valuation $v (\mu \leftrightarrow v)$, it holds

$$(\varPi, v) \models \sigma(gp_1) \text{ and } \text{dom}(v) = \text{var}(\sigma(gp_1)).$$

By Definition 3.5, it holds

$$(\varPi, v) \models \sigma(gp_1 \text{ minus } gp_2).$$

By Lemma 3.4, from $\text{dom}(v) = \text{var}(\sigma(gp_1))$, it holds $\text{dom}(v) = \sigma(\text{var}(gp_1))$, and then by Definition 2.7 it holds $\text{dom}(v) = \sigma(\varphi(gp_1 \text{ minus } gp_2))$. Therefore, by Lemma 3.4, it holds $\text{dom}(v) = \sigma(\varphi(gp_1 \text{ minus } gp_2))$.

$(\Leftarrow)$ By Definition 3.5, from

$$\text{dom}(v) = \sigma(\varphi(gp_1 \text{ minus } gp_2)),$$

it holds

$$(\varPi, v) \models \sigma(gp_1 \text{ minus } gp_2).$$

Applying the induction hypothesis on the valuation $v$ and its corresponding mapping $\mu (\mu \leftrightarrow v)$, it holds $\mu \notin [gp_1]_c$. Therefore, by Lemma 3.4, it holds $\text{dom}(\mu) = \varphi(gp_1)$. For all $\mu_2 \in [gp_2]_c$, by Lemma 3.4, it holds $\text{dom}(\mu_2) = \varphi(gp_2)$. Therefore, from $\text{var}(\varphi(gp_1)) \cap \text{var}(\varphi(gp_2)) = \emptyset$, for all $\mu_2 \in [gp_2]_c$, it holds

$$\text{dom}(\mu) \cap \text{dom}(\mu_2) = \emptyset.$$  

As it holds $\mu \in [gp_1]_c$ and for all $\mu_2 \in [gp_2]_c$ it holds $\text{dom}(\mu) \cap \text{dom}(\mu_2) = \emptyset$, then, by Definition 2.9

$$\mu \in [gp_1]_c \setminus [gp_2]_c.$$  

Then, by Definition 2.15, it holds

$$\mu \notin \{gp_1 \text{ minus } gp_2\}_c.$$  

Otherwise:

$(\Rightarrow)$ By Definition 2.15, from

$$\mu \notin [gp_1]_c \setminus [gp_2]_c,$$

it holds $\mu \notin \{gp_1 \text{ minus } gp_2\}_c$. By Definition 2.9, it holds $\mu \notin \{gp_1 \text{ minus } gp_2\}_c$. By induction hypothesis, from $\mu \notin \{gp_1 \text{ minus } gp_2\}_c$, for mapping $\mu$ and a corresponding valuation $v (\mu \leftrightarrow v)$, it holds

$$(\varPi, v) \models \sigma(gp_1) \text{ and } \text{dom}(v) = \text{var}(\sigma(gp_1)).$$

Let us prove by contraposition that it holds $(\varPi, v) \models \forall \varphi \neg \sigma(gp_2)$, i.e. let us assume

$$(\varPi, v) \not\models \exists \varphi \sigma(gp_2).$$

This means that we can extend the valuation $v$ to all the variables from $\text{var}(\sigma(gp_2))$ (by adding values to the variables from $\varphi$), and then restrict it only to the variables from $\text{var}(\sigma(gp_2))$ (by removing variables from $\text{var}(\sigma(gp_2))$ that are not in $\text{var}(\sigma(gp_2))$). Let $v_2$ denote such valuation. Then, it holds

$$(\varPi, v_2) \models \sigma(gp_2).$$

Note that it holds $v_2 \approx v$ by construction. From the induction hypothesis applied on $v_2$ and a mapping $\mu_2$ such that $\mu_2 \leftrightarrow v_2$, it holds $\mu_2 \in [gp_2]_c$. By Lemma 4.3, from $\mu \leftrightarrow v$, $\mu_2 \leftrightarrow v_2$ and $v \approx v_2$, it holds $\mu \approx \mu_2$. By Lemma 2.1, from $\mu \notin [gp_1]_c$ and $\mu_2 \in [gp_2]_c$, it holds $\text{dom}(\mu_1) = \varphi(gp_1)$ and $\text{dom}(\mu_2) = \varphi(gp_2)$. From $\text{var}(gp_1) \cap \text{var}(gp_2) = \emptyset$, it holds $\text{dom}(\mu_1) \cap \text{dom}(\mu_2) = \emptyset$. This is a contradiction with nonexistence of a mapping compatible with $\mu$, from $[gp_2]_c$ such that $\text{dom}(\mu_1) \cap \text{dom}(\mu_2) = \emptyset$. Therefore, it holds:

$$(\varPi, v) \models \forall \varphi \neg \sigma(gp_2).$$
and then by Definition 4.2
\[(\mathcal{D}, v) \models \sigma(gp_1) \land \forall \overline{x} \neg \sigma(gp_2).\]

By Definition 3.5, it holds
\[(\mathcal{D}, v) \models \sigma(gp_1 \text{ minus } gp_2).\]

By Lemma 3.4, from \(dom(v) = \text{var}(\sigma(gp_1))\), it holds \(dom(v) = \text{var}(\sigma(gp_1))\), and then by Definition 2.7 it holds \(dom(v) = \text{var}(\sigma(gp_1 \text{ minus } gp_2))\). Therefore, by Lemma 3.4, it holds \(dom(v) = \text{var}(\sigma(gp_1 \text{ minus } gp_2))\).

\((\Leftarrow)\) By Definition 3.5, from \(dom(v) = \text{var}(\sigma(gp_1 \text{ minus } gp_2))\), and
\[(\mathcal{D}, v) \models \sigma(gp_1 \text{ minus } gp_2)\]
it holds
\[dom(v) = \text{var}(\sigma(gp_1) \land \forall \overline{x} \neg \sigma(gp_2)),\]
and
\[(\mathcal{D}, v) \models \sigma(gp_1) \land \forall \overline{x} \neg \sigma(gp_2),\]
i.e. by Definition 4.2,
\[(\mathcal{D}, v) \models \sigma(gp_1) \text{ and } (\mathcal{D}, v) \models \forall \overline{x} \neg \sigma(gp_2).\]

By Definition 3.7, as \(\overline{x}\) denotes all variables that appear in \(\sigma(gp_2)\), but not in \(\sigma(gp_1)\), it holds \(dom(v) = \text{var}(\sigma(gp_1))\). Applying the induction hypothesis on the valuation \(v\) and its corresponding mapping \(\mu(\mu \leftrightarrow v)\), it holds \(\mu \in [gp_1]_g^0\).

It should also be proved that there does not exist a \(\mu_2\) such that \(\mu_2 \in [gp_2]_g^0\), \(\mu_2 \approx \mu\) and \(dom(\mu) \neq \emptyset\). This can be proved by contraposition, i.e. let us assume the existence of such a mapping \(\mu_2\).

By induction hypothesis, applied on the mapping \(\mu_2\) and a valuation \(v_2\) such that \(\mu_2 \leftrightarrow v_2\), it holds
\[(\mathcal{D}, v_2) \models \sigma(gp_2).\]

By Lemma 4.3, from \(\mu_2 \approx \mu, \mu \leftrightarrow v\) and \(\mu_2 \leftrightarrow v_2\), it holds \(v_2 \approx v\). Then, \(v \cup v_2\) is a well defined valuation, denoted by \(v'\). As \(v'\) is an extension of \(v_2\), from \((\mathcal{D}, v_2) \models \sigma(gp_2)\), it holds
\[(\mathcal{D}, v') \models \sigma(gp_2).\]

As \(v'\) is an extension of \(v\), from \((\mathcal{D}, v) \models \forall \overline{x} \neg \sigma(gp_2)\), it holds
\[(\mathcal{D}, v') \models \forall \overline{x} \neg \sigma(gp_2).\]

Therefore, there is an extension \(v''\) of \(v'\), such that
\[(\mathcal{D}, v'') \models \neg \sigma(gp_2).\]

Also, as \(v''\) is an extension of \(v'\), from \((\mathcal{D}, v') \models \sigma(gp_2)\), it holds
\[(\mathcal{D}, v'') \models \sigma(gp_2).\]

Therefore, \((\mathcal{D}, v'')\) is a model of formula \(\sigma(gp_2)\) and its negation as well, which is a contradiction.

As it holds \(\mu \in [gp_1]_g^0\) and it holds that there does not exist \(\mu_2\) such that \(\mu_2 \in [gp_2]_g^0\) and \(\mu_2 \approx \mu\), then, by Definition 2.9
\[\mu \in [gp_1]_g^0 - [gp_2]_g^0.\]

Then, by Definition 2.15, it holds
\[\mu \in [\text{graph } x (gp_1)]_g^0.\]

\(\Rightarrow\) By Definition 2.15, from
\[\mu \in [\text{graph } x (gp_1)]_g^0,\]
it holds
\[\mu \in \bigcup_{i \in \text{names}(0)} [gp_1]_{gr(i)}^0 \cup \{\mu_{x \leftarrow i}\}.\]

Therefore, \(I_n \neq \emptyset\) and there is an IRI \(i\) from \(\text{names}(0)\) such that
\[\mu \in [gp_1]_{gr(i)}^0 \cup \{\mu_{x \leftarrow i}\}.\]

By Definition 2.9, there exists mapping \(\mu_1\) such that \(\mu_1 = \mu_1 \cup \mu_{x \leftarrow i}, \mu_1 \in [gp_1]_{gr(i)}^0\) and \(\mu_1 \approx \mu_{x \leftarrow i}\). Applying the induction hypothesis on the mapping \(\mu_1\) and a valuation \(v_1\) such that \(\mu_1 \leftrightarrow v_1\), it holds
\[(\mathcal{D}, v_1) \models \sigma_{gr(i)}(gp_1).\]

By Lemma 4.4, from \(\mu = \mu_1 \cup \mu_{x \leftarrow i}, \mu \leftrightarrow v_1, \mu_1 \leftrightarrow v_1\) and \(\mu_{x \leftarrow i} \leftrightarrow v_{x \leftarrow i}\), it holds \(v = v_1 \cup v_{x \leftarrow i}\). Therefore, \(v \supseteq v_1\), and it holds
\[(\mathcal{D}, v) \models \sigma(\mu) = \sigma(i).\]

Then, by Definition 4.2, it holds:
\[(\mathcal{D}, v) \models \sigma_{gr(i)}(gp_1) \land \sigma(x) = \sigma(i).\]

By Definition 3.5, from \(i \in \text{names}(0), for i = (\sigma_i)^{-1}(i)\), it holds \(i \in I_n\). Therefore, it holds
\[(\mathcal{D}, v) \models \bigwedge_{i \in I_n} \left(\sigma_{gr(i)}((\sigma_i)^{-1}(i))(gp_1) \land \sigma(x) = i\right),\]
and, by Definition 3.5,
\[(\mathcal{D}, v) \models \sigma(\text{graph } x (gp_1)).\]

\((\Leftarrow)\) From \(dom(v) = \text{var}(\sigma(\text{graph } x (gp_1)))\) and
By Lemma 4.6, from the second satisfiability and \( \mu \leftrightarrow v \), it holds \( \mu(x) = \{1\} \), i.e. by Definition 2.13, it holds \( \mu(x) = 1 \). We consider two cases:

- If \( \sigma(x) \in \text{var}(\sigma(gp_1)) \), i.e. \( \text{dom}(v) = \text{var}(\sigma(gp_1)) \). By induction hypothesis, from the first satisfiability and \( \mu \leftrightarrow v \), it holds \( \mu \in \text{gp}_1 \cap \sigma(x) \in (i) \). Therefore, \( \mu = \mu \cup \mu_{v=1} \).

- If \( \sigma(x) \notin \text{var}(\sigma(gp_1)) \), let \( v' \) denotes a restriction of \( v \) to the set \( \text{var}(\sigma(gp_1)) \). Then, it holds \( (\sigma, v') \in \sigma(\text{gp}_{\sigma(0)}) \). Then, for a mapping \( \mu' \) such that \( \mu' \leftrightarrow v' \), by induction hypothesis, it holds \( \mu' \in \text{gp}_1 \cap \sigma(x) \). By Definition 4.4, from \( \mu \leftrightarrow v \), \( \mu' \leftrightarrow v' \) and \( \mathcal{V} \leq \mu \), it holds \( \mu' \leq \mu \).

Therefore, by Definition 4.2, it holds

\[
(\sigma, v) \vdash \sigma(\text{graph} \times \{gp_1\}),
\]

Putting both cases together, by Definition 2.9 it holds

\[
\mu \in \text{gp}_1 \cap \sigma(x) \in (i),
\]

Finally, by Definition 2.15 it holds that

\[
\mu \in \text{graph} \times \{gp_1\}^D_0.
\]
By Lemma 4.8, for a mapping \( \mu' \) such that \( \mu' \leftrightarrow \nu' \), it holds that \( \mu' \in \llbracket qpat \rrbracket_{df(f)}^D \). By Definitions 2.10 and 4.4, from \( \mu \leftrightarrow \nu, \mu' \leftrightarrow \nu' \) and \( \nu \leq \nu' \), it holds \( \mu \preceq \mu' \). Therefore, \( \mu \) is a restriction of \( \mu' \) with the following domain:

\[
dom(\mu) = (\sigma_1)^{-1}(\dom(\nu))
\]

(by Def 4.4, as \( \mu \leftrightarrow \nu \))

\[
= (\sigma_1)^{-1}(\overline{\nu})
\]

(by Lemma 3.5)

\[
= \overline{\nu}
\]

(by Lemma 2.2)

\[
= \var(qpat) \cap \overline{\nu}
\]

(by Lemma 2.1)

\[
= \var(\nu) \cap \overline{\nu}
\]

Therefore, by Definition 2.10, \( \mu = \mu' \) and by Definitions 2.11 and 2.16

\[
\mu \in \llbracket [\nu] \rrbracket^D.
\]

\( \square \)

Note that a mapping \( \mu \) is always considered as a mapping from \( \forall \) to \( \forall \). However, in Lemma 4.9, the actual \( \dom(\mu) \) is a subset of \( \nu \) and it does not contain blank nodes (according to Definition 2.16).

**Lemma 4.10** \( \Theta \) is valid if and only if \( q_1 \) is unsatisfiable.

**Proof.** (\( \Rightarrow \)) Assume that \( q_1 \) is satisfiable. By Definition 2.21, there exist a dataset \( D \) and a mapping \( \mu \) such that \( \mu \in \llbracket [\nu] \rrbracket^D \). Then, by Lemma 4.9, for a valuation \( \nu \) such that \( \mu \leftrightarrow \nu \), it holds

\[
(\mathcal{D}, \nu) \models \Phi_1(\overline{\nu}_1),
\]

where \( \overline{\nu}_1 \) is a set of relevant variables of \( q_1 \) and formula \( \Phi_1(\overline{\nu}_1) \) corresponds to \( q_1 \). Therefore, it holds

\[
(\mathcal{D}, \nu) \models \exists \overline{\nu}_1 \Phi_1(\overline{\nu}_1).
\]

By Definition 3.8, from the validity of \( \Theta \), it holds

\[
(\mathcal{D}, \nu) \models \neg(\exists \overline{\nu}_1 \Phi_1(\overline{\nu}_1)).
\]

Therefore, \( (\mathcal{D}, \nu) \) is a model of a formula and its negation, that is not possible, i.e. \( q_1 \) is unsatisfiable.

(\( \Leftarrow \)) Assume that \( \Theta \) is not valid. Therefore, there exists an \( L \)-structure \( \mathcal{D}' = (D', I') \), such that \( \mathcal{D}' \models \neg \Theta \). By Definition 3.8, \( \neg \Theta \) is equal to

\[
\exists \overline{\nu}_1 \Phi_1(\overline{\nu}_1).
\]

Note that this formula is a sentence (all variables are quantified). By Lemma 4.2, there exists a dataset \( D \) and the corresponding \( L \)-structure \( \mathcal{D} = (D, I^D) \), such that

\[
\mathcal{D} \models \exists \overline{\nu}_1 \Phi_1(\overline{\nu}_1).
\]

Therefore, there exists a valuation \( \nu \) defined on \( \overline{\nu}_1 \), such that

\[
(\mathcal{D}, \nu) \models \Phi_1(\overline{\nu}_1),
\]

i.e. by Definition 4.2,

\[
(\mathcal{D}, \nu) \models \Phi_1(\overline{\nu}_1).
\]

By Lemma 3.5, it holds \( \dom(\nu) = \var(\Phi_1(\overline{\nu}_1)) \). Therefore, by Lemma 4.9, for a mapping \( \mu \) such that \( \mu \leftrightarrow \nu \), it holds

\[
\mu \in \llbracket [\nu] \rrbracket^D.
\]

This is a contradiction with the unsatisfiability of \( q_1 \). Therefore, \( \Theta \) is valid.

**4.2. Soundness**

The following soundness theorem states that if the Procedure 3.1 claims that two queries are in the containment relation, they really are.

**Theorem 4.11** (Soundness) Let \( q_1 \) and \( q_2 \) be queries, \( \Theta \) be a formula generated from \( q_1 \) (Def 3.8) and \( \Psi \) be a formula generated from \( q_1 \) and \( q_2 \) (Def 3.9). It holds \( q_1 \sqsubseteq q_2 \) if one of the following conditions is satisfied:

1. \( \Theta \) is valid, or
2. \( q_1 \sim q_2 \) holds and \( \Psi \) is valid.

**Proof.** Case (1): \( \Theta \) is valid.

By Lemma 4.10, \( q_1 \) is unsatisfiable. By Definition 2.21, \( \llbracket [\nu] \rrbracket^D \) is an empty set for any dataset \( D \). Therefore, it holds \( \llbracket [\nu] \rrbracket^D \subseteq \llbracket [\nu] \rrbracket^D \), i.e. by Definition 2.18, \( q_1 \sqsubseteq q_2 \).

Case (2): It holds \( q_1 \sim q_2 \) and \( \Psi \) is valid.

Let \( D \) be any dataset, and \( \mu \) a mapping such that

\[
\mu \in \llbracket [\nu] \rrbracket^D.
\]

By Lemma 4.9, for a valuation \( \nu \) such that \( \mu \leftrightarrow \nu \), it holds

\[
(\mathcal{D}, \nu) \models \Phi_1(\overline{\nu}_1) \text{ and } \dom(\nu) = \var(\Phi_1(\overline{\nu}_1)).
\]

By Lemma 3.5, it holds \( \dom(\nu) = \var(\Phi_1(\overline{\nu}_1)) \). Therefore, by Lemma 4.9, for a mapping \( \mu \) such that \( \mu \leftrightarrow \nu \), it holds

\[
\mu \in \llbracket [\nu] \rrbracket^D.
\]
Soundness theorem for subsumption (dual to Theorem 4.11) can be proved as well. It validates the results of Procedure 3.2.

**Theorem 4.12** (Soundness for subsumption) Let $Q_1$ and $Q_2$ be queries, $\Theta$ be a formula generated from $Q_1$ (Def 3.8) and $\Psi$ be a formula generated from $Q_1$ and $Q_2$ (Def 3.9). It holds $Q_1 \subseteq Q_2$ if one of the following conditions is satisfied:

(1) $\Theta$ is valid, or
(2) $Q_1 \sim Q_2$ holds and $\Psi$ is valid.

**Proof.** This proof is equal to the proof of Theorem 4.11, from beginning to the applying Lemma 3.6. Instead of it, in the subsumption case, we apply its dual lemma, i.e. Lemma 3.7, and from $Q_1 \sim Q_2$ it holds

$$\overline{\nu_1} \subseteq \text{var}(\text{qpat}_2) \cap \overline{\nu_2},$$

i.e.

$$\sigma(\overline{\nu_1}) \subseteq \sigma(\text{var}(\text{qpat}_2) \cap \overline{\nu_2}),$$

and, by Definition 3.1, it holds

$$\sigma(\overline{\nu_1}) \subseteq \sigma(\overline{\text{var}(\text{qpat}_2)}) \cap \sigma(\overline{\nu_2}).$$

Therefore, by Lemma 3.4, it holds

$$\overline{\nu_1} \subseteq \sigma(\text{var}(\text{qpat}_2)) \cap \overline{\nu_2}. \quad (6)$$

The rest of the proof follows.

Let $v'$ denote any extension of $v$, such that $\text{dom}(v') = \text{dom}(v) \cup (\text{var}(\text{qpat}_2) \setminus \overline{\nu_2})$. By Definitions 3.6 and 3.9, as $\Psi$ is valid, it holds

$$(\text{D}, v'') \vdash \forall \overline{\nu_1} (\Phi_1(\overline{\nu_1}) \Rightarrow \exists \overline{\nu_2} \sigma^{df(2)}(\text{qpat}_2)).$$

As $v'$ is defined on all free variables from formula $\Phi_1(\overline{\nu_1}) \Rightarrow \exists \overline{\nu_2} \sigma^{df(2)}(\text{qpat}_2)$, it holds:

$$(\text{D}, v') \vdash \Phi_1(\overline{\nu_1}) \Rightarrow \exists \overline{\nu_2} \sigma^{df(2)}(\text{qpat}_2).$$

From $(\text{D}, v') \vdash \Phi_1(\overline{\nu_1})$, as $v' \geq v$ it holds $(\text{D}, v') \vdash \Phi_1(\overline{\nu_1})$. Therefore, it holds

$$(\text{D}, v') \vdash \exists \overline{\nu_2} \sigma^{df(2)}(\text{qpat}_2).$$

Then, there exists $v''$, an extension of $v'$, such that $\text{dom}(v'') = \text{dom}(v') \cup \overline{\nu_2}$ and

$$(\text{D}, v'') \vdash \sigma^{df(2)}(\text{qpat}_2).$$

Also: $\text{dom}(v'') = \text{dom}(v) \cup (\text{var}(\text{qpat}_2) \setminus \overline{\nu_2}) \cup \overline{\nu_2}$.

Therefore, by Theorem 4.13, for a mapping $\mu''$ such that $\mu'' \leftrightarrow v''$, it holds

$$\mu'' \notin [qpat_2]^{2}_{df(2)}.$$ 

By Definitions 2.10 and 4.4, from $\mu \leftrightarrow v$, $\mu'' \leftrightarrow v''$ and $v \preceq v''$, it holds $\mu \preceq \mu''$. Therefore, $\mu$ and $\mu''$ are restrictions of $\mu''$. By Definition 3.10, as $Q_1 \sim Q_2$, it holds $\overline{\nu_1} \subseteq \overline{\nu_2}$. Then, by Lemma 2.2, it holds $\overline{\text{var}(\text{qpat}_1)} \cap \overline{\nu_1} \subseteq \overline{\text{var}(\text{qpat}_2)} \cap \overline{\nu_2}$, i.e. $\text{dom}(\mu) \subseteq \text{dom}(\mu'')$ and $\mu \preceq \mu''$. Therefore, for each element of the set $[Q_1]^{2}$ there exists an extension that belongs to the set $[Q_2]^{2}$ for any dataset $D$. By Definition 2.19, it holds $Q_1 \subseteq Q_2$. \hfill \Box

4.3. Completeness

The following completeness theorem states that for each pair of queries satisfying the containment relation, the Procedure 3.1 will confirm that.

**Theorem 4.13** (Completeness) Let $Q_1$ and $Q_2$ be queries, $\Theta$ be a formula generated from $Q_1$ (Def 3.8) and $\Psi$ be a formula generated from $Q_1$ and $Q_2$ (Def 3.9). If $Q_1 \subseteq Q_2$, then one of the following conditions is satisfied:

(1) $\Theta$ is valid, or
(2) $Q_1 \sim Q_2$ holds and $\Psi$ is valid.

**Proof.** Case (1): $Q_1$ is unsatisfiable.

By Lemma 4.10, from the unsatisfiability of $Q_1$, $\Theta$ is valid.

Case (2): $Q_1$ is satisfiable.

We prove $Q_1 \sim Q_2$ and the validity of $\Psi$. As query $Q_1$ is satisfiable, there exists a dataset $D$ and a mapping $\mu$ such that $\mu \in [Q_1]^{D}$. By Definition 2.18, from $Q_1 \subseteq Q_2$, it holds $[Q_1]^{D} \subseteq [Q_2]^{D}$. Therefore, it holds $\mu \in [Q_2]^{D}$. By Definition 2.17, it holds $\text{dom}(\mu) = \overline{\nu_1}$ and $\text{dom}(\mu) = \overline{\nu_2}$. Therefore, it holds $\overline{\nu_1} = \overline{\nu_2}$, i.e. by Definition 3.10, $Q_1 \sim Q_2$.

Let us assume that $\Psi$ is not valid. Therefore, there exists an $L$-structure $\mathcal{D'} = (D', \mathcal{T'}'^D)$, such that $\mathcal{D'} \vdash \neg \Psi$. By Lemma 3.8, as $Q_1 \sim Q_2$ and $Q_2$ does not contain projections, $\neg \Psi$ is equal to

$$\neg (\forall \overline{\nu_1} (\Phi_1(\overline{\nu_1}) \Rightarrow \sigma^{df(2)}(\text{qpat}_2)), \quad (7)$$

where $D_2$ is a query dataset of $Q_2$, i.e.

$$\exists \overline{\nu_1} (\Phi_1(\overline{\nu_1}) \wedge \sigma^{df(2)}(\text{qpat}_2)). \quad (8)$$

M. Spasić and M. Vujošević Janičić: Soundness and Completeness of SPARQL Query Containment Solver SpeCS
Note that this formula is a sentence (all variables are quantified). By Lemma 4.2, there exists a dataset \( D \) and the corresponding \( L \)-structure \( \mathfrak{D} = (D, I^D) \), such that
\[
\mathfrak{D} \models \exists \overline{r_1} \left( \Phi_1(\overline{r_1}) \land \neg \sigma^{df(\varphi_2)}(\text{qpat}_2) \right).
\] (9)
Therefore, there exists a valuation \( v \) defined on \( \overline{r_1} \), such that
\[
(\mathfrak{D}, v) \models \Phi_1(\overline{r_1}) \land \neg \sigma^{df(\varphi_2)}(\text{qpat}_2),
\] i.e. by Definition 4.2,
\[
(\mathfrak{D}, v) \models \Phi_1(\overline{r_1}) \quad \text{and} \quad (\mathfrak{D}, v) \not\models \neg \sigma^{df(\varphi_2)}(\text{qpat}_2).
\] (11)
By Lemma 3.5, it holds \( \text{dom}(v) = \text{var}(\Phi_1(\overline{r_1})) \). Therefore, by Lemma 4.9, for a mapping \( \mu \) such that \( \mu \leftrightarrow v \), it holds
\[
\mu \in \left[\left[ q_1 \right]\right]^D.
\]
By Lemma 3.6, as \( q_1 \sim q_2 \) and \( q_2 \) does not contain projection, it holds
\[
\overline{r_1} = \text{var}(\text{qpat}_2).
\]
Therefore, it holds
\[
\sigma(\overline{r_1}) = \sigma(\text{var}(\text{qpat}_2)),
\]
i.e. by Lemma 3.4,
\[
\overline{r_1} = \text{var}(\sigma(\text{qpat}_2)).
\]
Therefore, \( \text{dom}(v) \) is equal to \( \text{var}(\sigma(\text{qpat}_2)) \), and then, by Lemma 4.8, it holds
\[
\mu \not\in \left[\left[ \text{qpat}_2 \right]\right]^{\sigma(\text{df}(\varphi_2))}.
\]
Then, by Definition 2.16, as query \( q_2 \) does not have projections, it holds
\[
\mu \not\in \left[\left[ q_2 \right]\right]^D.
\]
We conclude that \( \left[\left[ q_1 \right]\right]^D \subseteq \left[\left[ q_2 \right]\right]^D \) does not hold, i.e. by Definition 2.18, it does not hold \( q_1 \subseteq q_2 \). This is a contradiction with the theorem assumption. Therefore, \( \Psi \) is valid. \( \square \)

Completeness theorem for subsumption (dual to Theorem 4.13) can be proved as well.

**Theorem 4.14 (Completeness for subsumption)** Let \( q_1 \) and \( q_2 \) be queries, \( \Theta \) be a formula generated from \( q_1 \) (Def 3.8) and \( \Psi \) be a formula generated from \( q_1 \) and \( q_2 \) (Def 3.9). If \( q_1 \sqsubseteq q_2 \) holds, then one of the following conditions is satisfied:

\begin{enumerate}
\item \( \Theta \) is valid, or
\item \( q_1 \sim q_2 \) holds and \( \Psi \) is valid.
\end{enumerate}

**Proof.** In the case when \( q_1 \) is unsatisfiable, the proof is exactly the same as the proof of Theorem 4.13. Otherwise, the proof of \( q_1 \sim q_2 \) is done in the similar way as in the proof of Theorem 4.13: By Definition 2.21, as query \( q_1 \) is satisfiable, there exists a dataset \( D \) and a mapping \( \mu \) such that \( \mu \in \left[\left[ q_1 \right]\right]^D \). By Definition 2.19, from \( q_1 \sqsubseteq q_2 \), there exists an extension \( \mu' \) of \( \mu \) such that \( \mu' \in \left[\left[ q_2 \right]\right]^D \). By Definition 2.17, it holds \( \text{dom}(\mu) = \overline{r_1} \) and \( \text{dom}(\mu') = \overline{r_2} \). Therefore, it holds \( \overline{r_1} \subseteq \overline{r_2} \), i.e. by Definition 3.10, \( q_1 \sim q_2 \).

In the rest of the proof, until the applying Lemma 3.6, the following changes are necessary. Instead of satisfiabilities (7), (8), (9), (10) and (11), in the subsumption case, as \( q_2 \) can contain projections, it holds respectively:
\[
\overline{r_1} \not\models \left( \overline{r_1} \models \Phi_1(\overline{r_1}) \Rightarrow \exists \overline{r_2} \sigma^{df(\varphi_2)}(\text{qpat}_2) \right)
\]
\[
\overline{r_1} \models \Phi_1(\overline{r_1}) \land \neg \exists \overline{r_2} \sigma^{df(\varphi_2)}(\text{qpat}_2)
\]
\[
\mathfrak{D} \models \exists \overline{r_1} \left( \Phi_1(\overline{r_1}) \land \forall \overline{r_2} \neg \sigma^{df(\varphi_2)}(\text{qpat}_2) \right)
\]
\[
(\mathfrak{D}, v) \not\models \left( \Phi_1(\overline{r_1}) \land \forall \overline{r_2} \neg \sigma^{df(\varphi_2)}(\text{qpat}_2) \right)
\]
\[
(\mathfrak{D}, v') \not\models \Phi_2(\overline{r_2}),
\]
i.e. by Definition 3.6,
\[
(\mathfrak{D}, v') \not\models \exists \overline{r_2} \sigma^{df(\varphi_2)}(\text{qpat}_2),
\]
where \( \overline{r_2} = \text{var}(\sigma^{df(\varphi_2)}(\text{qpat}_2)) \setminus \overline{r_1} \). Note that by Definitions 2.10 and 4.4, as \( \mu \leftrightarrow v \), \( \mu' \leftrightarrow v' \), and \( \mu \leq \mu' \), it holds \( v \leq v' \). There exists an extension \( v'' \) of \( v' \) to the variables from \( \overline{r_2} \), such that
\[
(\mathfrak{D}, v'') \models \sigma^{df(\varphi_2)}(\text{qpat}_2),
\]
but also
\[
(\mathfrak{D}, v'') \not\models \exists \overline{r_2} \sigma^{df(\varphi_2)}(\text{qpat}_2).
\]
From \( v \leq v' \) and \( v' \leq v'' \), \( v \) is a restriction of \( v'' \). Therefore, it holds
\[
(\mathfrak{D}, v'') \not\models \forall \overline{r_2} \neg \sigma^{df(\varphi_2)}(\text{qpat}_2).
\]
This is a contradiction, as \( (\mathfrak{D}, v'') \) is a model of both, a formula and its negation. Therefore, \( \Psi \) is valid. \( \square \)

5. Dealing with Non-Conjunctive Queries

The containment problem of non-conjunctive queries, or more precisely queries containing the operators union (Section 5.1) and optional (Section 5.2), and containing subqueries (Section 5.3) can be transformed and/or reduced to the containment problem over (a set of) conjunctive queries presented in Section 3. Having correctness proved for conjunctive queries (Section 4), the query containment problem for non-conjunctive queries presented in this section is reduced to a sound and complete approach.

5.1. Operator union

Dealing with queries containing the union operator requires a transformation of their graph patterns into a special equivalent form where the union operator can appear only outside the scope of other operators. This condition is presented in the following definition.
Definition 5.1 (Simple normal form) A simple normal form of a graph pattern is a graph pattern containing \( n \) union-free graph patterns \( gp^1 (1 \leq i \leq n) \) connected with \( n - 1 \) union operators:

\[
gp^1 \cup \cdots \cup \gp^n
\]

Each graph pattern can be reduced to an equivalent pattern in a simple normal form following a standard DNF-style expansion [43].

Lemma 5.1 For queries \( q_1 \) and \( q_2 \), of the form select * over \( qpat_1 \) and select * over \( qpat_2 \), respectively, where query patterns \( qpat_1 \) and \( qpat_2 \) are in simple normal form consisting of union-free graph patterns \( gp^i_1 (1 \leq i \leq m) \) and \( gp^j_2 (1 \leq j \leq n) \) respectively, queries \( q_1^i \) (1 \( \leq i \leq m \)) and \( q_2^j \) (1 \( \leq j \leq n \)) are defined as:

\[
q_1^i = \text{select } \overline{dv} (\gp^i_1), \\
q_2^j = \text{select } * (\gp^j_2),
\]

where from and from named clauses of queries \( q_1 \) and \( q_2 \) are propagated to the queries \( q_1^i \) and \( q_2^j \), respectively. It holds:

\[
q_1 \sqsubseteq q_2 \quad \text{if and only if} \quad \text{for each } i, 1 \leq i \leq m \text{ there exists } j, 1 \leq j \leq n \text{ such that } q_1^i \sqsubseteq q_2^j.
\]

Proof. \((\Rightarrow)\) In order to prove this direction of lemma, for any \( i = 1..m \), there should be \( j = 1..n \) such that \( q_1^i \sqsubseteq q_2^j \). Assume that

\[
\mu \in [q_1]^D,
\]

for some \( i, i \in \{1, \ldots, m\} \) and some dataset \( D \). By Definitions 2.11 and 2.16, there exists a mapping \( \mu' \), such that \( \mu = \mu'_D \), and

\[
\mu' \in [gp^i_1 \cup \cdots \cup gp^n_2]^{d_1}_{df(\{q_1\})}.
\]

where \( d_1 \) is a query dataset specified by the from and from named clauses of query \( q_1^i \), and also of every query \( q_2 \). By Definition 2.15, it holds

\[
\mu' \in [gp^i_1 \cup \cdots \cup gp^n_2]^{d_1}_{df(\{q_1\})}.
\]

By Definitions 2.11 and 2.16, it holds

\[
\mu \in [q_1]^D,
\]

and also, as \( q_1 \sqsubseteq q_2 \),

\[
\mu \in [q_2]^D.
\]

By Definitions 2.11 and 2.16, as there is no projections in \( q_2 \),

\[
\mu \in [gp^i_2 \cup \cdots \cup gp^n_2]^{d_2}_{df(\{q_2\})},
\]

where \( d_2 \) is a query dataset specified by the from and from named clauses of query \( q_2 \), and also of every query \( q_2 \). By Definition 2.15, it holds

\[
\mu \in [gp^i_2 \cup \cdots \cup gp^n_2]^{d_2}_{df(\{q_2\})}.
\]

\((\Leftarrow)\) Let \( \mu \) be a mapping, and \( D \) a dataset. Assume that

\[
\mu \in [q_1]^D.
\]

By Definitions 2.11 and 2.16, there exists a mapping \( \mu' \), such that \( \mu = \mu'_D \), and

\[
\mu' \in [gp^i_1 \cup \cdots \cup gp^n_2]^{d_1}_{df(\{q_1\})},
\]

where \( d_1 \) is a query dataset specified by the from and from named clauses of query \( q_1 \), and also of every query \( q_1^i \), \( i = 1..m \). By Definition 2.15, there exists an index \( i, i \in \{1, \ldots, m\} \), such that \( \mu' \in [gp^i_1]^{d_1}_{df(\{q_1\})} \). By Definitions 2.11 and 2.16, it holds

\[
\mu \in [q_1]^D.
\]

By the assumption, there exists an index \( j, j \in \{1, \ldots, n\} \), such that

\[
\mu \in [q_2]^D.
\]

By Definitions 2.11 and 2.16, as there is no projections in \( q_2 \),

\[
\mu \in [gp^i_2 \cup \cdots \cup gp^n_2]^{d_2}_{df(\{q_2\})},
\]

i.e. by Definitions 2.11 and 2.16,

\[
\mu \in [q_2]^D.
\]

\( \square \)

Following Lemma 5.1, the query containment problem of queries \( q_1 \) and \( q_2 \) can be reduced to the checks if for each \( i \) \((1 \leq i \leq m) \) there exists \( j \) \((1 \leq j \leq n) \) such that \( q_1^i \sqsubseteq q_2^j \). The soundness and the completeness of the proposed modeling are a direct consequence of this lemma.

For the query subsumption problem, a weaker form of Lemma 5.1 is needed, considering relation \( \subseteq \) instead of \( \sqsubseteq \), whose proof is analogous.

5.2. Operator optional

All graph patterns explained so far demand to be matched completely. However, it is very useful to have possibility to add an information to the solution where the information is available, but do not reject the solution because some part of the query pattern does not match. Optional matching provides this facility.

For efficient usage of operator optional in practice, it is necessary to consider only well-designed graph patterns [45, 44, 43].

Definition 5.2 (Well-designed pattern) A pattern \( pat \) is well-designed if for all its subpatterns of the form \( gp_1 \) optional \( gp_2 \), it holds that all variables appearing in \( gp_2 \), but not in \( gp_1 \), cannot appear in \( pat \) anywhere else except in \( gp_2 \).
Proof. This is a direct corollary of Definitions 2.9 and 2.15.

Note that a graph pattern being introduced by a previously described reduction of a well-designed optional pattern is, by Definition 5.2, also a well-designed pattern.

5.3. Subqueries as Graph Patterns

Subqueries allow embedding queries within other queries in order to facilitate a composition of new queries and a reuse of existing queries. Subqueries are simpler than queries because they cannot contain from and from named clauses, and are used in a place of a graph pattern [2]. During the evaluation of a query, the inner most subquery is evaluated first (according to Definition 2.15), its results are projected up to the outer query, and only distinguished variables are visible (in scope) to the outer query (according to Definition 2.7).

In this section, we present a subquery elimination procedure, i.e. reduction of a query to an equivalent query without a subquery. We prove the correctness of this procedure. In order to simplify the proofs, we assume that queries do not contain graph, diff, and minus operators. If these operators are present, the subquery elimination is also possible, as the expressive power of the SPARQL language with and without subqueries is the same [29].

Definition 5.3 presents a special form of a graph pattern that simplifies reducing a graph pattern with a subquery into an equivalent query (by Definition 2.20) without the subquery.

Definition 5.3 (filter-normal form) A graph pattern gp is in filter-normal form if:

(i) gp is a triple pattern,

(ii) gp is \( q_{sa} \) and the query pattern qpat has of \( q_{sa} \) is in filter-normal form,

(iii) gp is \( q_{sa} \) and the query pattern qpat has of \( q_{sa} \) is in filter-normal form,

(iv) gp is \( q_{sa} \) and the query pattern qpat has of \( q_{sa} \) is in filter-normal form.

The following lemma shows that any graph pattern (considered so far) can be reduced to a graph pattern in filter-normal form.

Lemma 5.3 Let \( Q \) be an RDF dataset and \( G \) a graph within \( Q \). For any graph pattern \( gp \) there exists a graph pattern \( gp' \) in filter-normal form such that:

\[ [gp']^o_{filter} = [gp]^o. \]

Proof. The lemma is proved by induction over graph pattern gp.

\( gp \) is tp

By Definition 5.3, \( gp \) is in filter-normal form.

\( gp \) is \( gp_1 \cdot gp_2 \)

By Definition 2.15, it holds:

\[ [gp_1]_{filter}^o \bowtie [gp_2]^o. \]

By induction hypothesis, for \( gp_1 \) and \( gp_2 \) there exist graph patterns \( gp_1' \) and \( gp_2' \) in filter-normal form such that

\[ [gp_1']^o_1 \bowtie [gp_2']^o_2. \]

Case 1: \( gp_1' \) and \( gp_2' \) are of the form (i), (ii) or (iii).

Let \( gp' \) denote \( gp_1' \cdot gp_2' \). Then, by Definition 2.15, it holds:

\[ [gp']^o_1 \bowtie [gp_1']^o_2 \bowtie [gp_2']^o. \]

i.e.

\[ [gp']^o_1 \bowtie [gp_1']^o_2 \bowtie [gp_2']^o. \]

Therefore, it holds \( [gp']^o_1 \bowtie [gp_1']^o_2 \bowtie [gp_2']^o \) and by Definition 5.3, \( gp' \) is in filter-normal form.

Case 2: \( gp_1' \) has the form (iv), i.e.

\[ gp_1' = gp_1' \bowtie \text{filter} R_1, \]

while \( gp_2' \) is of the form (i), (ii) or (iii).

Then, it holds:

\[ [gp_1']^o_1 \bowtie [gp_2']^o_2 = [gp_1']^o_1 \bowtie [gp_1']^o_2 \bowtie [gp_2']^o. \]

By Definition 2.15, it holds \( \mu \bowtie R_1 \), then for its extension \( \mu \bowtie R_1 \), where \( \mu_1 \bowtie R_1 \) and \( \mu_2 \bowtie R_1 \). Therefore, it holds:

\[ [gp_1']^o_1 \bowtie [gp_2']^o_2 = \{ \mu \bowtie R_1 \}. \]

By Definition 2.15, it holds \( \mu \bowtie R_1 \), then for its extension \( \mu \bowtie R_1 \), where \( \mu_1 \bowtie R_1 \) and \( \mu_2 \bowtie R_1 \). Therefore, it holds:

\[ [gp_1']^o_1 \bowtie [gp_2']^o_2 = \{ \mu \bowtie R_1 \}. \]

Case 3: \( gp_2' \) is in the form (iv), while \( gp_1' \) is in the form (i), (ii) or (iii).

This case is reduced to the previous one because of the commutativity of the SPARQL operator . [43]. i.e.

\[ [gp_1']^o_1 \bowtie [gp_2']^o_2 = [gp_1']^o_1 \bowtie [gp_2']^o. \]
Case 4: \( gp_1' \) and \( gp_2' \) are in the form (iv), i.e. in the form \( gp_1'' \) filter \( R_1 \) and \( gp_2'' \) filter \( R_2 \), respectively. Applying construction of a normal form from the cases 2 and 3, we get:
\[
\llbracket gp \rrbracket^0 = \llbracket (gp_1'', gp_2'') \text{ filter } R_1 \text{ filter } R_2 \rrbracket^0,
\]
i.e. by Definition 2.14, it holds
\[
\llbracket gp \rrbracket^0 = \llbracket (gp_1'', gp_2'') \text{ filter } R_1 \& R_2 \rrbracket^0.
\]

By Definition 5.3, \( (gp_1'', gp_2'') \) filter \( R_1 \& R_2 \) is in the filter-normal form.

Note that all the variables appearing in filter clause (cases 2, 3 and 4) appear also in the graph pattern in front of it. This is a consequence of the same feature that holds for the initial graph patterns, while this transformation preserves it.

\( gp \) is \( gp_1 \) filter \( R \)

By Definition 2.15, it holds
\[
\llbracket gp \rrbracket^0 = \{ \mu \in \llbracket gp_1 \rrbracket^0 \mid \mu \models R \}. \]

By induction hypothesis, for \( gp_1 \) there exists a graph pattern \( gp_1' \) in the filter-normal form such that
\[
\llbracket gp_1 \rrbracket^0 = \llbracket gp_1' \rrbracket^0.
\]

Therefore, it holds
\[
\llbracket gp \rrbracket^0 = \{ \mu \in \llbracket gp_1' \rrbracket^0 \mid \mu \models R \},
\]
i.e. by Definition 2.15,
\[
\llbracket gp \rrbracket^0 = \llbracket gp_1' \text{ filter } R \rrbracket^0.
\]

If \( gp_1' \) is in the form (i), (ii) or (iii), let \( gp' \) denote \( gp_1' \) filter \( R \). Then, it holds \( \llbracket gp \rrbracket^0 = \llbracket gp' \rrbracket^0 \) and by Definition 5.3, \( gp \) is in the filter-normal form.

If \( gp_1' \) is in the form (iv), i.e. \( gp_1'' \) filter \( R_1 \), let \( gp' \) denote \( gp_1'' \) filter \( R_1 \& R \). By Definition 2.14, it holds
\[
\llbracket gp_1'' \text{ filter } R_1 \text{ filter } R_2 \rrbracket^0 = \llbracket gp_1'' \text{ filter } R_1 \& R \rrbracket^0.
\]

Then, it holds \( \llbracket gp \rrbracket^0 = \llbracket gp_1' \rrbracket^0 \) and by Definition 5.3, \( gp' \) is in filter-normal form.

By induction hypothesis, for \( gp_1 \) there exists a graph pattern \( gp_1' \) in filter-normal form such that
\[
\llbracket gp \rrbracket^0 = \llbracket gp_1' \rrbracket^0.
\]

Therefore, by Definition 2.15,
\[
\llbracket gp \rrbracket^0 = \llbracket gp_1' \rrbracket^0,
\]
where \( gp \) is in filter-normal form.

By induction hypothesis, for \( qpat_{sq} \) there exists a graph pattern \( qpat_{sq}' \) in filter-normal form such that
\[
\llbracket qpat_{sq} \rrbracket^0 = \llbracket qpat_{sq}' \rrbracket^0.
\]

Let \( q_{sq}' \) be a query obtained from \( q_{sq} \) by switching the query pattern \( qpat_{sq} \) with \( qpat_{sq}' \). Therefore, by Definition 2.16, it holds
\[
\llbracket q_{sq} \rrbracket^0 = \llbracket q_{sq}' \rrbracket^0,
\]
i.e. by Definition 2.15,
\[
\llbracket qpat \rrbracket^0 = \llbracket (q_{sq}') \rrbracket^0.
\]

By Definition 5.3, \( (q_{sq}') \) is in filter-normal form.

\( \square \)

An example of a graph pattern \( gp \) not satisfying filter-normal form is given in Figure 6 (left), while its equivalent pattern \( gp' \) in filter-normal form is given in the same figure (right).

**Figure 6:** Reducing the graph pattern \( gp \) to its equivalent pattern \( gp' \) in the filter-normal form

**Lemma 5.4** Let \( D \) be an RDF dataset and \( \sigma \) a graph within \( D \).

For a graph pattern \( gp \) that contains a subquery \( (Q_{sq}) \), there exists a graph pattern \( gp' \) such that it holds \( \llbracket gp \rrbracket^0 = \llbracket gp' \rrbracket^0 \) and \( gp' \) has one of the following forms:

- \( (Q_{sq}') \),
- \( (Q_{sq}') \) filter \( R \),
- \( gp_1 \cdot (Q_{sq}') \).
\[ \text{gp}_1.(Q'_{sq}) \text{ filter } R \]

where \( \text{gp}_1 \) is in filter-normal form (i), (ii) or (iii) and \( (Q'_{sq}) \) is in filter-normal.

**Proof.** A proof of this lemma is a direct consequence of Definition 5.3, Lemma 5.3 and the associativity and commutativity of the SPARQL operator . [43]

Some variables in a query can be renamed, while the evaluation of the transformed query remains the same, as stated in the following lemma.

**Lemma 5.5** Let \( D \) be an RDF dataset. Let \( Q \) be a query, \( qpat \) and \( \overline{dv} \) its query pattern and its set of distinguished variables, respectively. Let \( qpat' \) be a graph pattern obtained from \( qpat \) by renaming all the variables from \( \text{var}(qpat) \setminus \overline{dv} \), while introducing fresh new ones. Let \( Q' \) be a query obtained from \( Q \) by changing its query pattern \( qpat \) with \( qpat' \). Then it holds:

\[
\llbracket Q \rrbracket^D = \llbracket Q' \rrbracket^D.
\]

**Proof.** By Definition 2.16, it holds

\[
\llbracket Q \rrbracket^D = \Pi_{\overline{dv}}(\llbracket qpat \rrbracket^D_{df(0)}),
\]
i.e. by Definition 2.11,

\[
\llbracket Q \rrbracket^D = \{ \mu_{\overline{dv}} \mid \mu \in \llbracket qpat \rrbracket^D_{df(0)} \}.
\]

By the construction of \( qpat' \), it holds

\[
\{ \mu_{\overline{dv}} \mid \mu \in \llbracket qpat \rrbracket^D_{df(0)} \} = \{ \mu'_{\overline{dv}} \mid \mu' \in \llbracket qpat' \rrbracket^D_{df(0)} \},
\]
as, by Definition 2.10, mappings from \( \llbracket qpat \rrbracket^D_{df(0)} \) and from \( \llbracket qpat' \rrbracket^D_{df(0)} \) match on variables from \( \overline{dv} \). Therefore, it holds

\[
\llbracket Q \rrbracket^D = \{ \mu'_{\overline{dv}} \mid \mu' \in \llbracket qpat' \rrbracket^D_{df(0)} \}.
\]

By Definition 2.11 it holds

\[
\llbracket Q \rrbracket^D = \Pi_{\overline{dv}}(\llbracket qpat' \rrbracket^D_{df(0)}).
\]

By Definition 2.16 and the construction of the query \( Q' \), it holds

\[
\llbracket Q \rrbracket^D = \llbracket Q' \rrbracket^D.
\]

\[ \square \]

Any query \( Q \) containing a subquery \( Q_{sq} \) can be reduced to an equivalent query \( Q' \) that does not contain this subquery. The construction of such query \( Q' \) is given in the proof of the following lemma.

**Lemma 5.6** Let \( D \) be an RDF dataset. Let \( Q \) be a query such that its query pattern contains a subquery \( (Q_{sq}) \). Then, there exists a graph pattern \( \text{gp}_{sq} \) such that it is not a subquery and if it is used within \( Q \) instead of \( (Q_{sq}) \), forming a new query \( Q' \), it holds \( \llbracket Q \rrbracket^D = \llbracket Q' \rrbracket^D \).

**Proof.** According to Lemma 5.4, there exists a query pattern that is in a form defined by the lemma, and that is equivalent to the original query pattern of the query \( Q \). Therefore, we can assume that the query \( Q \) already contains a query pattern in such form.

We prove the case when the query pattern \( qpat \) of the query \( Q \) is equal to \( \text{gp}_1.(Q_{sq}) \) filter \( R \). All other cases are simpler and can be proved in analogous way.

In the subquery \( Q_{sq} \) of the query pattern \( qpat \), we can rename all the variables that are not distinguished \( \llbracket \text{var}(qpat_{sq}) \setminus \overline{dv_{sq}} \rrbracket \), by introducing fresh new variables that do not appear in the outer query \( Q \) (anywhere else except in the subquery \( Q_{sq} \)). This step is possible as a set of variables \( \overline{v} \) is countable.

After this renaming, by Lemma 5.5, it holds that the evaluation of the subquery \( Q'_{sq} \) that contains renamed variables, has not been changed compared to the subquery \( Q_{sq} \). Let \( qpat'_{sq} \) denote the query pattern of \( Q'_{sq} \).

The set of variables from the outer query \( Q \) includes distinguished variables \( \overline{dv} \) and variables of the graph pattern \( \text{gp}_1 \), i.e. \( \text{var}(\text{gp}_1) \). Therefore, it holds:

\[
\text{var}(qpat'_{sq}) \cap (\text{var}(\text{gp}_1) \cup \overline{dv}) \subseteq \overline{dv_{sq}}
\]
i.e.

\[
\text{var}(qpat'_{sq}) \cap \text{var}(\text{gp}_1) \subseteq \overline{dv_{sq}}
\]
and

\[
\text{var}(qpat'_{sq}) \cap \overline{dv} \subseteq \overline{dv_{sq}}
\]

Note that it also holds \( \text{var}(R) \subseteq \text{var}(\text{gp}_1) \cup \overline{dv_{sq}} \), i.e. by Definition 2.7,

\[
\text{var}(R) \subseteq \text{var}(\text{gp}_1) \cup \overline{dv_{sq}}.
\]

Let \( qpat' \) be a query pattern such that in query pattern \( qpat \), the subquery \( Q_{sq} \) is replaced with \( \{ qpat'_{sq} \} \). Let \( Q' \) be a query such that in the query \( Q \), its query pattern \( qpat \) is replaced with \( qpat' \). Let us prove that \( \llbracket Q \rrbracket^D = \llbracket Q' \rrbracket^D \).

\( \langle \rangle \) Let \( \mu \) be a mapping such that \( \mu \in \llbracket Q \rrbracket^D \). By Definition 2.16, there exists a mapping \( \mu' \), such that

\[
\mu = \mu'_{\overline{dv}} \quad \text{and} \quad \mu' \in \llbracket qpat' \rrbracket^D_{df(0)}.
\]

Therefore, by Definitions 2.15 and 2.15, it holds

\[
\mu' \in \llbracket \text{gp}_1 \rrbracket^D_{df(0)} \quad \text{and} \quad \mu' \models R.
\]

By Definition 2.9, there exist compatible mappings \( \mu_1 \) and \( \mu'' \) such that

\[
\mu'' = \mu_1 \cup \mu'' \quad \text{and} \quad \mu'' \models R.
\]

By Definition 2.16, there exists a mapping \( \mu''' \) such that

\[
\mu''' = \mu'_{\overline{dv_{sq}}}. \quad \text{and} \quad \mu''' \in \llbracket qpat'_{sq} \rrbracket^D_{df(0)}.
\]

By Definition 2.8, from \( \mu_1 \models \mu'' \), it holds \( \mu_1 \models \mu''' \), as there is no variable in \( \text{var}(qpat'_{sq}) \) apart from \( \overline{dv_{sq}} \) that can appear in \( \text{var}(\text{gp}_1) \) because of (12). Let \( \mu^i \) denote mapping \( \mu_1 \cup \mu'' \). By Definition 2.9, it holds

\[
\mu^i \in \llbracket \text{gp}_1 \rrbracket^D_{df(0)} \quad \text{and} \quad \mu^i \models R.
\]
By Definition 2.10, as \( \mu^i = \mu_1 \cup \mu^{ii} \), \( \mu^j = \mu_1 \cup \mu^i \) and \( \mu^{ii} \geq \mu^i \), it holds \( \mu^{iv} \geq \mu^i \). Then, for \( \mu^{iv} \) as an extension of \( \mu^i \), from \( \mu^i \Vdash R \), it holds:

\[
\mu^{iv} \Vdash R.
\]

Therefore, by Definition 2.15, it holds

\[
\mu^{iv} \in \llbracket qpat' \rrbracket^{D}_{df(f)}.
\]

i.e. by Definition 2.16,

\[
\mu^{iv}_{\overline{dv}} \in \llbracket q' \rrbracket^{D}.
\]

Let us prove that \( \mu^{iv}_{\overline{dv}} = \mu \). By Definition 2.10, from \( \mu^{iv} \geq \mu^i \), it holds \( \mu^{iv}_{\overline{dv}} \geq \mu_{\overline{dv}} \), i.e. \( \mu^{iv}_{\overline{dv}} \geq \mu \). Also, it holds:

\[
dom(\mu) =
\]

(by Def 2.10)

\[
= \dom(\mu^i) \cap \overline{dv}
\]

(as \( \mu^i = \mu_1 \cup \mu^{ii} \))

\[
= (\dom(\mu_1) \cup \dom(\mu^{ii})) \cap \overline{dv}
\]

(by dist.of \( \cap \)

over \( U \))

\[
= (\dom(\mu)^i) \cap \overline{dv}
\]

(by Def 2.10)

\[
= (\dom(\mu_1) \cap \overline{dv}) \cup (\dom(\mu^{ii}) \cap \overline{dv})
\]

(by Lemma 2.1)

\[
= (\dom(\mu_1) \cap \overline{dv}) \cup (\var(qpat'_{sq}) \cap \overline{dv} \cap \overline{dv})
\]

(from (13))

\[
= (\dom(\mu_1) \cap \overline{dv}) \cup (\var(qpat'_{sq}) \cap \overline{dv} \cap \overline{dv})
\]

(by Lemma 2.1)

\[
= (\dom(\mu_1) \cap \overline{dv}) \cup (\var(qpat'_{sq}) \cap \overline{dv} \cap \overline{dv})
\]

(by dist.of \( \cap \) over \( U \))

\[
= (\dom(\mu_1) \cup \dom(\mu^{ii})) \cap \overline{dv}
\]

(as \( \mu^{iv} = \mu_1 \cup \mu^{ii} \))

\[
= \mu^{iv}_{\overline{dv}}
\]

(by Def 2.10)

\[
= \mu^{iv}_{\overline{dv}}
\]

Therefore, it holds \( \mu^{iv}_{\overline{dv}} = \mu \), and

\[
\mu \in \llbracket q' \rrbracket^{D}.
\]

(2) Let \( \mu \) be a mapping such that \( \mu \in \llbracket q' \rrbracket^{D} \). By Definition 2.16, there exists a mapping \( \mu' \), such that

\[
\mu = \mu^i_{\overline{dv}} \quad \text{and} \quad \mu' \in \llbracket qpat' \rrbracket^{D}_{df(f)}.
\]

Therefore, by Definition 2.15, it holds

\[
\mu' \in \llbracket gp_1 \rrbracket^{D}_{df(f)} \triangleright \llbracket qpat'_{sq} \rrbracket^{D}_{df(f)} \quad \text{and} \quad \mu' \Vdash R.
\]

By Definition 2.9, there exist compatible mappings \( \mu_1 \) and \( \mu^{ii} \) such that

\[
\mu' = \mu_1 \cup \mu^{ii} \quad \text{and} \quad \mu_1 \in \llbracket gp_1 \rrbracket^{D}_{df(f)} \quad \text{and} \quad \mu^{ii} \in \llbracket qpat' \rrbracket^{D}_{df(f)}.
\]

By Definition 2.16, it holds

\[
\mu^{ii}_{\overline{dv}} \in \llbracket q'_{sq} \rrbracket^{D}.
\]

By Definitions 2.8 and 2.10, as \( \mu_1 \simeq \mu^i \), it holds \( \mu_1 \simeq \mu^{ii}_{\overline{dv}} \). Let \( \mu^{ii} \) denote mapping \( \mu_1 \cup \mu^{ii}_{\overline{dv}} \). By Definition 2.9, it holds

\[
\mu^{ii} \in \llbracket gp_1 \rrbracket^{D}_{df(f)} \triangleright \llbracket q'_{sq} \rrbracket^{D}.
\]

By Definition 2.10, it holds \( \mu_1 \cup \mu^{ii} \simeq \mu_1 \cup \mu^{ii}_{\overline{dv}} \), i.e. \( \mu^i \geq \mu^{ii} \).

Note that \( \dom(\mu^i) = \dom(\mu_1) \cup \dom(\mu^{ii}) \) and \( \dom(\mu^{ii}) = \dom(\mu_1) \cup (\dom(\mu^{ii}) \cap \overline{dv}_{sq}) \), i.e. by Lemma 2.1,

\[
\dom(\mu^i) = \var(gp_1) \cup \var(qpat'_{sq}),
\]

\[
\dom(\mu^{ii}) = \var(gp_1) \cup (\var(qpat'_{sq}) \cap \overline{dv}_{sq}).
\]

Then,

\[
\dom(\mu^i) \setminus \dom(\mu^{ii}) = \var(qpat'_{sq}) \setminus \overline{dv}_{sq}.
\]

All of the variables from \( \dom(\mu^i) \setminus \dom(\mu^{ii}) \) are renamed, and they cannot belong to \( \var(r) \), because of (14). Therefore, from \( \mu^i \Vdash R \) it holds

\[
\mu^{ii} \Vdash R.
\]

Then, by Definition 2.16, it holds

\[
\mu^{ii}_{\overline{dv}} \in \llbracket q' \rrbracket^{D}.
\]

Let us prove that \( \mu^{ii}_{\overline{dv}} = \mu \). By Definition 2.10, from \( \mu^{ii} \leq \mu^i \), it holds \( \mu^{ii}_{\overline{dv}} \leq \mu_{\overline{dv}} \), i.e. \( \mu^{ii}_{\overline{dv}} \leq \mu \). Also, it holds:

\[
dom(\mu) =
\]

(by Def 2.10)

\[
= \dom(\mu^i) \cap \overline{dv}
\]

(as \( \mu^i = \mu_1 \cup \mu^{ii} \))

\[
= (\dom(\mu_1) \cup \dom(\mu^{ii})) \cap \overline{dv}
\]

(by dist.of \( \cap \)

over \( U \))

\[
= (\dom(\mu)^i) \cap \overline{dv}
\]

(by Def 2.10)

\[
= (\dom(\mu_1) \cap \overline{dv}) \cup (\dom(\mu^{ii}) \cap \overline{dv})
\]

(by Lemma 2.1)

\[
= (\dom(\mu_1) \cap \overline{dv}) \cup (\var(qpat'_{sq}) \cap \overline{dv} \cap \overline{dv})
\]

(from (13))

\[
= (\dom(\mu_1) \cap \overline{dv}) \cup (\var(qpat'_{sq}) \cap \overline{dv} \cap \overline{dv})
\]

(by Lemma 2.1)

\[
= (\dom(\mu_1) \cap \overline{dv}) \cup (\var(qpat'_{sq}) \cap \overline{dv} \cap \overline{dv})
\]

(by dist.of \( \cap \) over \( U \))

\[
= (\dom(\mu_1) \cup \dom(\mu^{ii})) \cap \overline{dv}
\]

(as \( \mu^{iv} = \mu_1 \cup \mu^{ii} \))

\[
= \mu^{iv}_{\overline{dv}}
\]

(by Def 2.10)

\[
= \mu^{iv}_{\overline{dv}}
\]

Therefore, it holds \( \mu^{ii}_{\overline{dv}} = \mu \), and

\[
\mu \in \llbracket q' \rrbracket^{D}.
\]
If a query $\varnothing$ contains more than one subquery, then the elimination of subqueries should be applied starting from the inner most subquery. This way, the containment problem concerning queries with subqueries is reduced to the containment problem of queries that do not contain subqueries.

For the containment problem, we assume that there is no subqueries in the query $Q_0$ or in cases where they are present in $Q_2$, all their variables are projected up to $Q_2$. Otherwise, the construction of an equivalent query $Q'_2$ from the proof of Lemma 5.6 would make projections present within $Q'_2$, leading to an undecidable problem. When considering the subsumption relation, this restriction is not necessary, as the query subsumption problem is decidable in cases when a super-query contains projections [45].

6. Conclusions and Further Work

In this paper we proved correctness of the approach described in [56] and implemented by a tool SPECS [54]. Correctness, i.e. soundness and completeness, are proved for modeling conjunctive queries for both containment and subsumption relation. Soundness and completeness proofs are extended to cover non-conjunctive queries containing operator union, optional, and subqueries in cases when well designed patterns and filter-normal form are considered.

There are several possible directions for further work: extending the language coverage while keeping the soundness and completeness of the approach if possible, considering containment and subsumption in the context of RDF SCHEMA, considering containment and subsumption when a wider set of axioms is included (e.g. $SHI$ axioms [28, 15] or code refactoring context [53]), applying a similar approach for different graph query languages (e.g. XPath [52, 41], GQL [25]), and making the presented proofs formal within a proof assistant [42, 8].

References

[1] Angles, R., Gutierrez, C., 2008. The Expressive Power of SPARQL, in: Proceedings of the 7th International Conference on The Semantic Web, Springer. pp. 114–129. doi: 10.1007/978-3-540-85664-1_8.
[2] Angles, R., Gutiérrez, C., 2011. Subqueries in SPARQL, in: Proceedings of the 5th Alberto Mendelzon International Workshop on Foundations of Data Management, Santiago, Chile, May, 2011, CEUR-WS.org. pp. 1–12. URL: http://ceur-ws.org/Vol-749/paper19.pdf.
[3] Angles, R., Gutiérrez, C., 2016. Negation in SPARQL, in: Proceedings of the 10th Alberto Mendelzon International Workshop on Foundations of Data Management, Panama City, Panama, 2016, CEUR-WS.org. pp. 1–10. URL: http://ceur-ws.org/Vol-1644/paper11.pdf.
[4] Angles, R., Gutiérrez, C., 2016. The Multiset Semantics of SPARQL Patterns, in: The Semantic Web – ISWC 2016. Springer. pp. 20–36. doi: 10.1007/978-3-319-45633-5_2.
[5] Arnas, M., Perez, J., 2011. Querying Semantic Web Data with SPARQL, in: Proceedings of the 13th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 305–316. doi: 10.1145/1989284.1989312.
[6] Auer, S., Lehmann, J., Ngomo, A.C.N., 2011. Introduction to Linked Data and Its Lifecycle on the Web, in: Reasoning Web. Semantic Technologies for the Web of Data. Springer. pp. 1–75. doi: 10.1007/978-3-642-23082-5_1.
[7] Barrett, C., Sebastiani, R., Seshia, S.A., Tinelli, C., 2009. Satisfiability Modulo Theories, in: Handbook of Satisfiability, IOS Press. pp. 825–885. doi: 10.5555/1558723.
[8] Bertot, Y., Castran, P., 2010. Interactive Theorem Proving and Program Development: Coq’Art The Calculus of Inductive Constructions. Springer. doi: 10.1007/978-3-662-07964-5.
[9] Calvanese, D., Giacomo, G.D., Lenzneri, M., Vardi, M.Y., 2000. Containment of Conjunctive Regular Path Queries With Inverse, in: Proceedings of the 7th International Conference on Principles of Knowledge Representation and Reasoning, Morgan Kaufmann. pp. 176–185. URL: https://dl.acm.org/doi/10.5555/3087811.3087932.
[10] Chandra, A.K., Merlin, P.M., 1977. Optimal Implementation of Conjunctive Queries in Relational Data Bases, in: STOC. ACM. pp. 77–90. doi: 10.1145/800185.803397.
[11] Chang, C.C., Keisler, H.J., 1976. Model Theory. Journal of Symbolic Logic 41, 697–699. doi: 10.2307/2272047.
[12] Chekol, M., Vujošević Janičić, M., 2011. Soundness and Completeness of the Approach if possible, considering containment and subsumption when a wider set of axioms is included (e.g. $SHI$ axioms [28, 15] or code refactoring context [53]), applying a similar approach for different graph query languages (e.g. XPath [52, 41], GQL [25]), and making the presented proofs formal within a proof assistant [42, 8].
