Superimposed Multidimensional Schema Proofs

This chapter is intended to provide formal proofs for some of the definitions and statements presented in Chapter 9. We first present basics of SPARQL algebra and semantics in Section 1. Afterward, in Section 2, we provide some formal proofs for definitions stated in Chapter 9 that are concerned with the SPARQL queries for MD data analysis using the superimposed MD model. Finally, we prove some features over the MD schema of a superimposed MD schema in Section 3.

1 SPARQL 1.1 Algebra

In this section, we provide some basic definitions of the SPARQL 1.1 query language [1] that are used throughout this dissertation. Whereas most of the presented definitions are derived from the W3C recommendation [1] and SPARQL literature [2]–[5], we occasionally present our own definitions that serve the purposes of this dissertation. For further details on SPARQL, we refer the reader to the W3C recommendation [1] as well as literature [2]–[5]. Note that some of the information in this section are duplicated from Chapter 9 for the sake of self-containment. Note also that whenever we refer to SPARQL, we mean SPARQL 1.1 [1].

1.1 SPARQL Preliminaries

In this section, some SPARQL preliminaries are presented.

- RDF Terms and Variables. The set of RDF terms $\mathcal{T}$ is defined as $\mathcal{T} = \mathcal{I} \cup \mathcal{B} \cup \mathcal{L}$, which is composed of the union of the sets of IRIs $\mathcal{I}$, blank nodes $\mathcal{B}$, and literals $\mathcal{L}$. The set of SPARQL variables is denoted as $\mathcal{V}$. The sets $\mathcal{I}$, $\mathcal{B}$, $\mathcal{L}$, and $\mathcal{V}$ are pairwise disjoint.
- RDF Triple. An RDF triple t is defined as an element of $(\mathcal{I} \cup \mathcal{B}) \times \mathcal{I} \times (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L})$.
- Triple Pattern. The set of triple patterns is defined as $\mathcal{TRP} = (\mathcal{I} \cup \mathcal{L} \cup \mathcal{V}) \times (\mathcal{I} \cup \mathcal{V}) \times (\mathcal{I} \cup \mathcal{L} \cup \mathcal{V})$. A triple pattern is defined as an element of $\mathcal{TRP}$. As in many approaches in the literature, e.g., [2] and [3], we do not consider the set of blank nodes $\mathcal{B}$ in formulating triple patterns.
- Basic Graph Pattern (BGP). A BGP is a set of triple patterns, i.e., an element of the set $2^{\mathcal{TRP}}$.
- Solution Mapping. Solution mappings are used in the evaluation of SPARQL. In particular, a solution mapping μ is a partial function that maps variables from $\mathcal{V}$ to RDF terms from $\mathcal{T}$, i.e., $\mu : \mathcal{V} \rightarrow \mathcal{T}$. The domain of a solution mapping μ, denoted as $dom(\mu)$ where $dom(\mu) \subseteq \mathcal{V}$, comprises the variables $v \in \mathcal{V}$ where $\mu(v)$ is defined.
- SPARQL Graph Patterns. We define $\mathbb{P}$ as the set of all SPARQL graph patterns. $\mathbb{P}$ is disjoint from $\mathcal{V}$ and $\mathcal{T}$ (refer to [3] and [1] for further details on graph patterns).
- SPARQL Graph Patterns of Mapping Queries. We define $\mathcal{P}$ as the set of SPARQL graph patterns allowed at the outermost level inside the projection in the mapping queries (see Definition 9.3). We require that for all $p \in \mathcal{P}$, it holds that p is of one of the patterns defined in Table 9.2. Whereas p itself must be one of the specified patterns, it may comprise any other graph patterns. This restriction facilitates the implementation of some optimization techniques (refer to Section 9.4.6). This restriction, however, could be relaxed to allow other patterns. We also require each pattern $p \in \mathcal{P}$ to be free of non-determinism (see [4] for further details).

- SPARQL Projection Queries. We define $\mathcal{Q}$ as the set of SPARQL projection queries. We require that for all $q \in \mathcal{Q}$, the following holds: (i) the pattern of q is projection query (Table 9.2) and (ii) q does not contain any solution modifier except for projection (refer to [1] and [5] for details on solution modifiers).

- RDF Data Set. An RDF graph is defined as a set of RDF triples. An RDF data set DS comprises one default RDF graph in addition to named graphs. For simplicity, we assume that an RDF data set comprises only the default graph, which we denote as G. We also assume that the default graph is the active graph (refer to [1] for more details).

- Evaluation. Given a pattern $p \in \mathbb{P}$ and a data set DS, $[\![p]\!]_{DS}$ denotes the evaluation of p against DS. Multisets of solution mappings are the main building block for the evaluation of patterns as well as for SPARQL algebra. In particular, a multiset is a pair $M = (\Omega, card)$ where: (i) Ω is a set of solution mappings (also called the base set of M [3]), and (ii) $card$ is a cardinality function that specifies the number of occurrences of solution mappings in Ω such that: if $\mu \in \Omega$ then $card(\mu) > 0$ and if $\mu \notin \Omega$ then $card(\mu) = 0$ [2], [3]. Given a multiset $M = (\Omega, card)$, we sometimes write $\mu \in M$ in order to denote that $\mu \in \Omega$.

- Pattern Variables. The function $vars : \mathbb{P} \rightarrow 2^{\mathcal{V}}$ maps a pattern to its *possible variables*, i.e., the variables that appear in $dom(\mu)$ of some solution mapping μ for the pattern evaluation against some data set [4], [2]. Note that determining if a variable is a member of the possible variables is undecidable in SPARQL 1.1 (refer to [4] for details). However, these variables are approximated, such as in [2], [4]. The function $cVars : \mathbb{P} \rightarrow 2^{\mathcal{V}}$ maps a pattern to its *certain variables*, i.e., the variables that appear in $dom(\mu)$ of any solution mapping μ for the pattern evaluation against any data set [4], [2]. Note that determining if a variable is a member of the certain variables is undecidable in SPARQL 1.1 (refer to [4] for details). However, these variables are approximated, such as in [2], [4].

- Compatibility. Given two solution mappings μ_1 and μ_2, we say that μ_1 and μ_2 are *compatible* if and only if: $\forall x \in dom(\mu_1) \cap dom(\mu_2)$, it holds that $\mu_1(x) = \mu_2(x)$ (we also say in this case that μ_1 and μ_2 agree on the variables in $dom(\mu_1) \cap dom(\mu_2)$ [2]). We denote the compatibility of two solution mapping μ_1 and μ_2 as $\mu_1 \sim \mu_2$. If $\mu_1 \sim \mu_2$, then we define their combination $\mu_1 \cup \mu_2$ as follows [4]: (i) $dom(\mu_1 \cup \mu_2) = dom(\mu_1) \cup dom(\mu_2)$, (ii) if $x \in dom(\mu_1)$, then $(\mu_1 \cup \mu_2)(x) = \mu_1(x)$, and (iii) if $x \in dom(\mu_2)/dom(\mu_1)$ then $(\mu_1 \cup \mu_2)(x) = \mu_2(x)$ [4].

1.2 Basic SPARQL Algebra Syntax and Semantics

This section provides some basic definitions of SPARQL algebra syntax and semantics. Before we proceed to present the syntax and semantics, we briefly discuss some relevant points.

First, it is worth noting that we always assume multisets of solution mappings. Whereas we state our definitions and propositions based on multisets of solution mappings, the result of evaluating a query in SPARQL 1.1 is a sequence of solution mappings [1]. The transformation

from SPARQL syntax to SPARQL algebra [1] involves a step that employs the function `ToList` to convert a multiset of solution mappings into a sequence of solution mappings (preserving members and cardinality) and, afterward, the various solution modifiers (`OrderBy`, `Project`, `Distinct`, `Reduced`, `Offset`, `Limit`) are applied (see [4], [5] for further details on solution modifiers). Nevertheless, before the results of a subquery are employed further in the evaluation of the enclosing query, the sequence of solution mappings yielded by evaluating the subquery is transformed into a multiset of solution mappings [1], [5]. Since order of solutions is irrelevant to the definitions presented in Chapter 9, the usage of multisets as presented does not affect our approach.

Second, note that symbols employed in the presented algebra syntax also denote algebra operators that are used in the evaluation of the algebra syntax. In particular, each of the symbols for projection π, duplicate elimination α, join $\bowtie$, and big join $\bigwedge$, introduced in Table 9.2, is a part of the syntax on the one hand, and denotes an algebra operator on the other hand. For example, whereas the join of two SPARQL graph patterns p_1 and p_2 is syntactically expressed using the join symbol $\bowtie$ (as $p_1 \bowtie p_2$), the semantics of the join is defined using the algebraic operator $\bowtie$ (as $[\![p_1 \bowtie p_2]\!]_{DS} = [\![p_1]\!]_{DS} \bowtie [\![p_2]\!]_{DS}$ where $[\![p]\!]_{DS}$ denotes the evaluation of the SPARQL graph pattern p against the data set DS).

In the following, we present some basic SPARQL algebra syntax and semantics that were employed in Chapter 9, assuming a data set DS.

- Join. Let $M_1 = (\Omega_1, card_1)$ and $M_2 = (\Omega_2, card_2)$ be multisets of solution mappings. The join of M_1 and M_2 results in a multiset and is defined as $M_1 \bowtie M_2 = (\Omega', card')$ where: (i) $\Omega' = \{\mu_1 \cup \mu_2 \mid \mu_1 \in \Omega_1, \mu_2 \in \Omega_2, \text{ and } \mu_1 \sim \mu_2\}$ [2], [4], and (ii) $card'(\mu) = \sum_{(\mu_1,\mu_2) \in \{(\mu_1^*,\mu_2^*) \in \Omega_1 \times \Omega_2 | \mu_1^* \sim \mu_2^* \wedge \mu = \mu_1^* \cup \mu_2^*\}} card_1(\mu_1) \times card_2(\mu_2)$ [2], [4]. The semantics of join is defined as $[\![p_1 \bowtie p_2]\!]_{DS} = [\![p_1]\!]_{DS} \bowtie [\![p_2]\!]_{DS}$. Note that $vars(p_1 \bowtie p_2) = vars(p_1) \cup vars(p_2)$ and that $cVars(p_1 \bowtie p_2) = cVars(p_1) \cup cVars(p_2)$ [4].

- Big join. Let M_1, ..., M_n, with $n \geq 3$, be multisets of solution mappings. The big join $\bigwedge$ of these multisets results in a multiset and is defined as $\bigwedge_{i=1}^{n} M_i = (...(M_1 \bowtie M_2)...) \bowtie M_n$. The semantics of big join is defined as $[\![\bigwedge_{i=1}^{n} p_i]\!]_{DS} = [\![(...(p_1 \bowtie p_2)...)]\!]_{DS} \bowtie [\![p_n]\!]_{DS}$. Note that it is possible to adapt the definition to allow the case where $n = 2$; in this case, big join amounts to normal join.

- Projection. Let $M = (\Omega, card)$ be a multiset of solution mappings, and S be a set of projection variables. The projection of M on S results in a multiset and is defined as $\pi_S(M) = (\Omega', card')$ where: (i) $\Omega' = \{\mu' \mid \exists \mu \in \Omega : dom(\mu') = S \cap dom(\mu) \wedge \mu' \sim \mu\}$ [4] (we write $\mu' = \mu_{|S}$ [3] when $\exists \mu \in \Omega : dom(\mu') = S \cap dom(\mu) \wedge \mu' \sim \mu$), and (ii) $card'(\mu) = \sum_{\mu' \in \{\mu^* \in \Omega \mid dom(\mu) = S \cap dom(\mu^*) \wedge \mu \sim \mu^*\}} card(\mu')$ [4]. The semantics of projection is defined as $[\![\pi_S(p)]\!]_{DS} = \pi_S([\![p]\!]_{DS})$. Note that $vars(\pi_S(p)) = S$ and that $cVars(\pi_S(p)) = cVars(p) \cap S$ [4].

- Duplicate Elimination. Let $M = (\Omega, card)$ be a multiset of solution mappings. The duplicate elimination of M results in a multiset and is defined as $\alpha(M) = (\Omega', card')$ where: (i) $\Omega' = \Omega$, i.e., the base set of M' is the same base set of M, and (ii) $card'(\mu) = 1$ for all $\mu \in \Omega$, i.e., the cardinality of each solution mapping in Ω becomes 1 [2]. The semantics of duplicate elimination is defined as $[\![\alpha(p)]\!]_{DS} = \alpha([\![p]\!]_{DS})$.

2 Multidimensional Data Analysis using SPARQL

in this section, we provide some formal proofs for definitions stated in Chapter 9 that are concerned with the SPARQL queries for MD data analysis using the superimposed MD model.

2.1 Auxiliary Propositions and Definitions

This section presents some auxiliary propositions and definitions that are used later to present proofs.

Whereas we state our definitions and propositions based on multisets of solution mappings, the evaluation of queries in SPARQL 1.1 results in sequences of solution mappings [1]. The transformation from SPARQL syntax to SPARQL algebra [1] involves a step that employs the function $toList$ to convert a multiset of solution mappings into a sequence of solution mappings (preserving elements and cardinality) and, afterward, the various solution modifiers (`OrderBy`, `Project`, `Distinct`, `Reduced`, `Offset`, `Limit`) are applied (see [4], [5] for further details on solution modifiers). However, a sequence of solution mappings resulting from evaluating a subquery is transformed into a multiset before the subquery is used further in the evaluation of the enclosing query [1]. Furthermore, order of solution mappings is not relevant to our approach. As a consequence, using multisets does not affect our results.

In the following we define equality of multisets and equivalence of queries, which allow to clearly state our following proofs. The first two definitions address (set) equality of multisets as well as (set) equivalence of patterns, similarly to [2], [4]. Note that (set) equality of multisets as well as (set) equivalence of patterns are transitive relationships.

Definition 1 (Multiset Equality)**.** Assume two multisets of solutions $M_1 = (\Omega_1, card_1)$, $M_2 = (\Omega_2, card_2)$. M_1 and M_2 are *equal*, denoted as $M_1 = M_2$, if and only if (i) $\Omega_1 = \Omega_2$, and (ii) $\forall x \in \Omega_1 : card_1(x) = card_2(x)$. M_1 and M_2 are *set-equal*, denoted as $M_1 \simeq M_2$, if and only if $\mu \in M_1 \Leftrightarrow \mu \in M_2$ (i.e., $\Omega_1 = \Omega_2$).

Definition 2 (SPARQL Equivalence)**.** Assume two SPARQL patterns p_1, $p_2 \in \mathcal{P}$. p_1 and p_2 are *equivalent*, denoted as $p_1 \equiv p_2$, if and only if for any dataset d, the following holds: $[\![p_1]\!]_d = [\![p_2]\!]_d$. p_1 and p_2 are *set-equivalent*, denoted as $p_1 \cong p_2$, if and only if for any dataset d, the following holds: $[\![p_1]\!]_d \simeq [\![p_2]\!]_d$.

A special type of SPARQL queries, certain disjoint queries, have desired properties that we will use later to state our proofs.

Definition 3 (Query Disjointness)**.** Two queries $q_1 = \pi_{S_1}(p_1) \in \mathcal{Q}$ and $q_2 = \pi_{S_2}(p_2) \in \mathcal{Q}$ are disjoint if an only if the possible variables of the patterns of q_1 and q_2 are disjoint unless they occur in the respective projection variables (similar to disjoint non-distinguished variables in [6]). Formally, $\forall q_1, q_2 \in \mathcal{Q}$, $\forall S_1, S_2 \subset \mathcal{V}$, $\forall p_1, p_2 \in \mathcal{P}$ such that $q_1 \neq q_2$, $q_1 = \pi_{S_1}(p_1)$, and $q_2 = \pi_{S_2}(p_2)$: q_1 and q_2 are disjoint if and only if $vars(p_1) \cap vars(p_2) = S_1 \cap S_2$.

Definition 4 (Certain Query)**.** A query $q = \pi_S(p) \in Q$ is a certain query if and only if $\emptyset \neq S \subseteq cVars(p)$.

Definition 5 (Certain Multiset)**.** A multiset of solution mappings $M = (\Omega, card)$ is certain if and only if $\exists S \neq \emptyset : (\forall \mu \in M : dom(\mu) = S)$. We denote the set of all certain multisets of solution mappings as $\mathbb{M}$.

Note that in case a multiset is empty, then it is certain. However the set S as per the definition is not unique. If a multiset is not empty, then S is unique.

Note that our mapping queries from Chapter 9 are certain and disjoint. The certainty of these queries makes it easier to infer how solution mappings resulting from the evaluation of certain queries (or some operations over these queries, e.g., join) look like; see for example Proposition 1, Proposition 4, Proposition 5, which are silently used in proving following propositions.

Notational Conventions. We use the following conventions in the following propositions and proofs.

- Given an MD path $(v_1, ..., v_n)$, we may denote the mapping query $\kappa(v_i)$ of v_i as q_i for notational simplicity.
- A pattern p_1 can be set-rewritten to p_2 means that p_1 and p_2 are set-equivalent.
- Given an equivalence ... $\Leftrightarrow$ ..., LEFT denotes the left-hand side and RIGHT denotes the right-hand side.

In the following, we provide some auxiliary propositions that help stating proofs later in this chapter.

Proposition 1 (Evaluation of Certain Queries)**.** The evaluation of a certain query q against a dataset d results in a certain multiset, i.e., $[\![q]\!]_d \in \mathbb{M}$. In particular, for a certain query $q = \pi_S(p) \in \mathcal{Q}$, it holds that $\forall \mu \in [\![q]\!]_d$: $dom(\mu) = S$.

Proof. The proof follows directly from the definitions of certain queries and certain multisets as well as certain and possible variables [2], [4]. In particular, for a certain query $q = \pi_S(p) \in \mathcal{Q}$, we know that $cVars(q) = S \cap cVars(p)$ (certain variables [4]). However, from the definition of our mapping queries, we know that $S \subseteq cVars(p)$, which leads to (i) $cVars(q) = S$. We know that (ii) $vars(q) = S$ (possible variables [4]). We know from the definitions of certain and possible variables [2], [4], that (iii) $\forall \mu \in [\![q]\!]_d : cVars(q) \subseteq dom(\mu) \subseteq vars(q)$. From (i), (ii), and (iii), it follows that $S \subseteq dom(\mu) \subseteq S$, which means that $dom(\mu) = S$. Note that S by definition is different from empty set $\emptyset$. □

Proposition 2 (Join Commutativitty)**.** Given two patterns $p_1, p_2 \in \mathcal{P}$, the following holds: $p_1 \bowtie p_2 = p_2 \bowtie p_1$ [2], [4], [7].

Proposition 3 (Join Associativity)**.** Given three patterns $p_1, p_2, p_3 \in \mathcal{P}$, the following holds: $(p_1 \bowtie p_2) \bowtie p_3 = p_1 \bowtie (p_2 \bowtie p_3)$ [2], [4], [7].

From now on, we may drop brackets when specifying joins as that does not lead to an ambiguity according to the previous proposition.

Proposition 4 (Join of Certain Queries)**.** Given the certain queries $q_1 = \pi_{S_1}(p_1), ..., q_m = \pi_{S_m}(p_m)$ with $m \geq 2$, then $q = \pi_{\bigcup_{i=1}^{m} S_i} \left(\bigwedge_{i=1}^{m} q_i \right)$ is a certain query.

Proof. Let us check the conditions for a certain query on the resulting query q. (i) $q \in \mathcal{Q}$ as q is of the form $\pi_S(p)$, where $S = \bigcup_{i=1}^{m} S_i$ and $p = \bigwedge_{i=1}^{m} q_i$ where $p = \left(\bigwedge_{i=1}^{m} q_i \right) \in \mathcal{P}$ (p is a `JOIN` pattern), and (ii) $S = \left(\bigcup_{i=1}^{m} S_i \right) \subseteq cVars(p)$ because, for any data set d, any solution mapping $\mu \in [\![p]\!]_d$ is of the form (join definition) $(..(\mu_1 \cup \mu_2)...) \cup \mu_m$ where $dom((..(\mu_1 \cup \mu_2)...) \cup \mu_m) = dom((..(\mu_1 \cup \mu_2)...)) \cup dom(\mu_m) = dom(\mu_1) \cup .... \cup dom(\mu_m)$. We know that for $1 \leq i \leq m$, it holds that $dom(\mu_i) = S_i$ (Proposition 1). Hence, $dom(\mu_1) \cup .... \cup dom(\mu_m) = S_1 \cup ... \cup S_m = \bigcup_{i=1}^{m} S_i$. □

Note also that the proof can be done using the definition of certain variables [2], [4].

Proposition 5 (Projection of Certain Queries)**.** Given the certain query $q = \pi_S(p)$ and the set $S' \subseteq S$, then $q' = \pi_{S'}(q)$ is a certain query.

Proof. Since q is certain, it follows that $S \subseteq cVars(p)$. We know from certain variables [2], [4] that $cVars(q) = S \cap cVars(p) = S$. Since $S' \subseteq S = cVars(q)$ and $q' \in \mathcal{Q}$ it follows by comparison to certain queries (Definition 4) that q' is certain query. □

Proposition 6 (Self Join)**.** Given a certain query $q \in \mathcal{Q}$, then $q \cong q \bowtie q$.

Proof. Given any data set d, let $M = [\![q]\!]_d$. The proof is done by double inclusion on $\forall \mu (\mu \in M \Leftrightarrow \mu \in M \bowtie M)$.

LEFT ⇒ RIGHT. It is obvious that for any solution mapping μ, it holds that $\mu \sim \mu$. Hence, and as $\mu \in M$ (LEFT), then $\mu \cup \mu \in M \bowtie M$ (definition of join) which leads to $\mu \in M \bowtie M$ (RIGHT) as $\mu \cup \mu = \mu$.

RIGHT ⇒ LEFT. From $\mu \in M \bowtie M$ (RIGHT), we derive that $\exists \mu_1 \in M, \exists \mu_2 \in M : (\mu = \mu_1 \cup \mu_2) \wedge (\mu_1 \sim \mu_2)$ (definition of join). As M is a certain multiset (Proposition 1), it is clear that $dom(\mu_1) = dom(\mu_2)$. As $\mu_1 \sim \mu_2$ and $dom(\mu_1) = dom(\mu_2)$, then μ_1 and μ_2 should agree on all their variables, which can be only the case when $\mu_1 = \mu_2$. Hence, $\mu = \mu_1 \cup \mu_2 = \mu_1 \cup \mu_1$ and, therefore, $\mu = \mu_1$. As $\mu_1 \in M$, that implies that $\mu \in M$ (LEFT). □

The previous proposition and its proof are similar to the the Rule $\widetilde{JIdem}$ with the incompatibility property [2].

Proposition 7 (Asterisk Projection)**.** Given a SPARQL pattern $p \in \mathcal{P}$, then $\pi_{vars(p)}(p) \cong p$.

The proof results directly from the definition of projection and the possible variables of a pattern as in [2].

Proposition 8 (MD Path Partition)**.** Given:

- a node-to-node MD path $(v_1, v_2, ..., v_{n-1}, v_n)$ such that:
  - $n \geq 5$, and;
  - $(\forall i : 1 \leq i \leq n) : \kappa(v_i) = \pi_{S_i}(p_i)$, and;
  - for every node v_i in the path: $S_i = \{s_i\}$.
- $m \in \mathbb{N}$ such that:
  - $1 < m < n$, and;
  - v_m is a node, and;

then: $(S_1 \cup ... \cup S_m) \cap (S_m \cup ... \cup S_n) = S_m$.

Proof. We know from the definition of our mapping queries that given an edge $v_i = (v_{i-1}, v_{i+1})$, then the projection variables of v_i, denoted as S_i, are defined as $S_i = \{s_{i-1}, s_{i+1}\}$ where s_{i-1} is the projection variable of v_{i-1}, and s_{i+1} is the projection variable of v_{i+1}. Hence, $(S_1 \cup ... \cup S_m) = \{s_1\} \cup \{s_1, s_3\} \cup ... \cup s_m = \{s_1, s_3, ..., s_m\}$. Note that s_1, s_3, ..., s_m are the projection variables of the nodes v_1, v_3, ..., v_m, respectively. Similarly, $(S_m \cup ... \cup S_n) = \{s_m\} \cup \{s_m, s_{m+2}\} \cup ... \cup \{s_n\}$. Note also that s_m, s_{m+2}, ..., s_n are the projection variables of the nodes v_m, v_{m+2}, ..., v_n, respectively. Note that MD path do not contain cycles or loops and, hence, $\{v_m\} = \{v_1, v_3, ..., v_m\} \cap \{v_m, v_{m+2}, ..., v_n\}$. We know from mapping queries definitions that nodes have pairwise disjoint projection variables. Hence, $\{s_1, s_3, ..., s_m\} \cap \{s_m, s_{m+2} \cup ... \cup s_n\} = \{s_m\}$. □

In the following propositions, we always assume that the evaluation of patterns occurs against an arbitrary data set d, i.e., whenever we write $[\![...]\!]$, we mean $[\![...]\!]_d$.

Proposition 9 (Projection Partition)**.** Given a node-to-node MD path $(v_1, ..., v_n)$, with $n \geq 5$ the following holds:

$$\forall a_1, \forall a_n :$$

$$\Bigg(\underbrace{\Bigg(\{s_1 \mapsto a_1, s_n \mapsto a_n\} \in [\![\pi_{\{s_1,s_n\}} \underbrace{\Big(\bigwedge_{i=1}^{n} q_i \Big)}_{L}]\!] \Bigg)}_{L_\pi} \Leftrightarrow$$

$$\Bigg(\forall m \text{ such that } (1 < m < n) \ \wedge \ v_m \text{ is a node} : \exists a_m : \Bigg($$

$$\{s_1 \mapsto a_1, s_m \mapsto a_m\} \in [\![\pi_{\{s_1,s_m\}} \underbrace{\Bigg(\bigwedge_{i=1}^{m} q_i \Bigg)}_{R_1}]\!] \wedge$$

$$\{s_m \mapsto a_m, s_n \mapsto a_n\} \in [\![\pi_{\{s_m,s_n\}} \underbrace{\Big(\bigwedge_{i=m}^{n} q_i \Big)}_{R_2}]\!] \Bigg) \Bigg) \Bigg)$$

Proof. The proof is done by double inclusion. We first make the following shortcuts. $L_\pi \coloneqq \pi_{\{s_1,s_n\}} \left(\bigwedge_{i=1}^{n} q_i \right)$, $L \coloneqq \bigwedge_{i=1}^{n} q_i$, $R_1 \coloneqq \bigwedge_{i=1}^{m} q_i$, and finally $R_2 \coloneqq \bigwedge_{i=m}^{n} q_i$.

LEFT $\Rightarrow$ RIGHT. We start by assuming that LEFT is correct and end with the correctness of RIGHT. Note that by expanding the big join of L, we know that $\forall m : (1 < m < n$ and m is an odd number$)$: $L = \left(\bigwedge_{i=1}^{m-1} q_i \right) \bowtie q_m \bowtie \left(\bigwedge_{i=m+1}^{n} q_i \right)$ and, hence, $L = \left(\bigwedge_{i=1}^{m-1} q_i \right) \bowtie \left(\bigwedge_{i=m+1}^{n} q_i \right) \bowtie q_m$ (Proposition 3 and Proposition 2). Previous L can be set-rewritten to $L \simeq \left(\left(\bigwedge_{i=1}^{m-1} q_i \right) \bowtie \left(\bigwedge_{i=m+1}^{n} q_i \right) \right) \bowtie (q_m \bowtie q_m)$ (Proposition 6). Furthermore, $L \simeq \left(\left(\bigwedge_{i=1}^{m-1} q_i \right) \bowtie q_m \right) \bowtie \left(q_m \bowtie \left(\bigwedge_{i=m+1}^{n} q_i \right) \right)$ (Proposition 3 and Proposition 2), which can, further, be written as $L \simeq \left(\bigwedge_{i=1}^{m} q_i \right) \bowtie \left(\bigwedge_{i=m}^{n} q_i \right)$ which is the same as $R_1 \bowtie R_2$. Hence[2], $\forall \mu = \{s_1 \mapsto a_1, s_n \mapsto a_n\} \in L_\pi$: $\exists \mu_1 = \{s_1 \mapsto a'_1, ..., s_m \mapsto a'_m\} \in [\![R_1]\!]$, $\exists \mu_2 = \{s_m \mapsto a''_m, ..., s_n \mapsto a''_n\} \in [\![R_2]\!]$ such that $\mu_1 \sim \mu_2$ (join definition), and $(\mu_1 \cup \mu_2)_{|\{s_1,s_n\}} = \{s_1 \mapsto a_1, s_n \mapsto a_n\}$ (projection definition), which implies that $a_1 = a'_1$ and $a_n = a''_n$, i.e., $\mu_1 = \{s_1 \mapsto a_1, ..., s_m \mapsto a'_m\}$ and $\mu_2 = \{s_m \mapsto a''_m, ..., s_n \mapsto a_n\}$. Note that $dom(\mu_1) \cap dom(\mu_2) = \{s_m\}$ (Proposition 8), and, as $\mu_1 \sim \mu_2$, we get that $\mu_1(s_m) = \mu_2(s_m)$ and, hence, $a'_m = a''_m$. Note that (i) $\mu_1 = \{s_1 \mapsto a_1, ..., s_m \mapsto a'_m\} \in [\![R_1]\!]$ implies that $\{s_1 \mapsto a_1, s_m \mapsto$

[2]Note that s_1 and s_n are always bound in any solution mapping in L_π since they are always bound in any solution mapping in L.

$a'_m\} \in [\![\pi_{\{s_1,s_m\}}(R_1)]\!]$ (projection definition), and (ii) $\mu_2 = \{s_m \mapsto a'_m, ..., s_n \mapsto a_n\} \in [\![R_2]\!]$ implies that $\{s_m \mapsto a'_m, s_n \mapsto a_n\} \in [\![\pi_{\{s_m,s_n\}}(R_2)]\!]$ (projection definition). From (i) and (ii) we get that LEFT $\Rightarrow$ RIGHT.

RIGHT $\Rightarrow$ LEFT. We start by assuming that RIGHT is correct and end with the correctness of LEFT. If RIGHT is correct, then we have from R_1 that $\exists \mu'_1 = \{s_1 \mapsto a_1, s_m \mapsto a_m\} \in [\![\pi_{\{s_1,s_m\}}(R_1)]\!]$. From the definition of projection, we get that: $\exists \mu_1 = \{s_1 \mapsto a_1, ..., s_m \mapsto a_m\} \in [\![R_1]\!] : \mu'_1 = \mu_{1|_{\{s_1,s_m\}}}$. Similarly, we know from R_2 that $\exists \mu'_2 = \{s_m \mapsto a_m, s_n \mapsto a_n\} \in [\![\pi_{\{s_m,s_n\}}(R_2)]\!]$. From the definition of projection, we get that $\exists \mu_2 = \{s_m \mapsto a_m, ..., s_n \mapsto a_n\} \in [\![R_2]\!] : \mu'_2 = \mu_{2|_{\{s_m,s_n\}}}$. Note that $dom(\mu_1) \cap dom(\mu_2) = \{s_m\}$ (Proposition 8) and that $\mu_1 \sim \mu_2$ as $\mu_1(s_m) = \mu_2(s_m) = a_m$. Hence, by applying join definition it follows that: **(i)** $\mu_1 \cup \mu_2 \in [\![R_1]\!] \bowtie [\![R_2]\!]$. By expanding the big join, $R_1 \bowtie R_2$ can be written as $R_1 \bowtie R_2 = \left(\bigwedge_{i=1}^{m-1} q_i\right) \bowtie q_m \bowtie q_m \bowtie \left(\bigwedge_{i=m+1}^{m} q_i\right)$ (Proposition 3 and Proposition 2). That can be written as in the following: $R_1 \bowtie R_2 = \left(\left(\bigwedge_{i=1}^{m-1} q_i\right) \bowtie \left(\bigwedge_{i=m+1}^{m} q_i\right)\right) \bowtie (q_m \bowtie q_m)$ (Proposition 3 and Proposition 2). We can eliminate one of the two q_m queries (Proposition 6), i.e., we set-rewrite $R_1 \bowtie R_2$ as follows $R_1 \bowtie R_2 \simeq \left(\left(\bigwedge_{i=1}^{m-1} q_i\right) \bowtie \left(\bigwedge_{i=m+1}^{m} q_i\right)\right) \bowtie q_m$, and, hence, by collapsing the big join, it follows that **(ii)** $R_1 \bowtie R_2 \simeq \bigwedge_{i=1}^{n} q_i$. From (i) and (ii), it follows that $\mu_1 \cup \mu_2 = \{s_1 \mapsto a_1, ..., s_m \mapsto a_m, ..., s_n \mapsto a_n\} \in [\![\bigwedge_{i=1}^{n} q_i]\!]$. It follows from projection definition that $\{s_1 \mapsto a_1, s_n \mapsto a_n\} \in [\![\pi_{\{s_1,s_n\}} \left(\bigwedge_{i=1}^{n} q_i\right)]\!]$ which proves that RIGHT $\Rightarrow$ LEFT. $\square$

2.2 Proofs of Definitions in Chapter 9

This section presents proofs for definitions of SPARQL queries for MD data analysis under the superimposed MD model.

2.2.1 SPARQL Query of a Multidimensional Path (Definition 9.11)

Let $v_1, v_m \in E_{mapped}$ such that $v_1 \triangleright v_m$ via the path $(v_1, ..., v_m)$. Let the set A be defined as $A = \{(a_1, a_m) \in \lambda(v_1) \times \lambda(v_m) \mid a_1 \triangleright a_m\}$. Finally, let the set B be defined as $B = \{(b_1, b_m) \in \mathcal{T} \times \mathcal{T} \mid \exists \mu = \{s_1 \mapsto b_1, s_m \mapsto b_m\} \in [\![q]\!]$ with $q = \alpha\left(\pi_{\{s_1,s_m\}}\left(\bigwedge_{i=1}^{m} \kappa(v_i)\right)\right)\}$ where $\kappa(v_1) = \pi_{\{s_1\}}(p_1)$ and $\kappa(v_m) = \pi_{\{s_m\}}(p_m)$. Then the following equality holds: $A = B$.

Proof. Note that we prove that each solution mapping in the evaluation of the query q results in a respective member in A, and that each member in A has a respective solution mapping in the evaluation of the query q.

The proof is done by double-inclusion and induction. We first prove that it is correct for $m = 3$ by double-inclusion. We start with $m = 3$ as this is the minimum length of an MD path (Definition 9.2). Then we proceed to the induction step where we prove, by double-inclusion, that if the relation is correct for m, then it is correct for $m + 2$ as the length of an MD path is increased by 2 at a time (adding an edge and a node). Note that $m = 3$ means that the path is (v_1, v_2, v_3) which can be, alternatively, written as $(v_1, (v_1, v_3), v_3)$. Let A_m and B_m where $m \in \mathbb{N}$ denote the sets A and B when the length of the considered path is m. Note that duplication elimination operator α can be safely dropped form the query q since B is a set, i.e., free of duplicates.

Correctness for $m = 3$. Which means that the MD path is (v_1, v_2, v_3) where $v_2 = (v_1, v_3)$ and the respective query is $\alpha(\pi_{\{s_1,s_3\}}\left(\kappa(v_1) \bowtie \kappa(v_2) \bowtie \kappa(v_3)\right)$. In particular, we want to prove that:

$$\forall a_1, \forall a_3 : \left(\left((a_1, a_3) \in A_3\right) \Leftrightarrow \left(\mu = \{s_1 \mapsto a_1, s_3 \mapsto a_3\} \in \underbrace{[\![\pi_{\{s_1,s_3\}}\left(\kappa(v_1) \bowtie \kappa(v_2) \bowtie \kappa(v_3)\right)]\!]}_{R}\right)\right) \tag{1}$$

LEFT $\Rightarrow$ RIGHT. We first make the following shortcut: $R \coloneqq \pi_{\{s_1,s_3\}}\kappa(v_1) \bowtie \kappa(v_2) \bowtie \kappa(v_3)$. From the definition of mapping queries (Definition 9.3), we know that R can be written as in the following: $R = \pi_{\{s_1,s_3\}}\left(\pi_{\{s_1\}}(p_1) \bowtie \pi_{\{s_1,s_3\}}(p_2) \bowtie \pi_{\{s_3\}}(p_3)\right)$. Respectively, R can be evaluated as follows: $[\![R]\!] = [\![\pi_{\{s_1,s_3\}}\left(\pi_{\{s_1\}}(p_1) \bowtie \pi_{\{s_1,s_3\}}(p_2) \bowtie \pi_{\{s_3\}}(p_3)\right)]\!]$. By evaluating projection and join, we can write: $[\![R]\!] = \pi_{\{s_1,s_3\}}\left([\![\pi_{\{s_1\}}(p_1)]\!] \bowtie [\![\pi_{\{s_1,s_3\}}(p_2)]\!] \bowtie [\![\pi_{\{s_3\}}(p_3)]\!]\right)$. Hence, the LEFT $\Rightarrow$

RIGHT direction of the relation 1 can be written as:

$$\forall a_1, \forall a_3 : \Bigg(\Big((a_1, a_3) \in A_3 \Big) \Rightarrow \Big(\mu = \{s_1 \mapsto a_1, s_3 \mapsto a_3\} \in \pi_{\{s_1,s_3\}} \Big([\![\pi_{\{s_1\}}(p_1)]\!] \bowtie [\![\pi_{\{s_1,s_3\}}(p_2)]\!] \bowtie [\![\pi_{\{s_3\}}(p_3)]\!] \Big) \Big) \Bigg) \tag{2}$$

The proof is done by starting form the left-hand side of Relation 2 and proceeding until we get the right-hand side. We know from the definition of connected instances (Definition 9.5) that:

$$\begin{gathered} a_1 \triangleright a_3 \Rightarrow \exists (a_1, (a_1, a_3), a_3) : \\ a_1 \in \lambda(v_1) \wedge \\ (a_1, a_3) \in \lambda((v_1, v_3)) \wedge \\ a_3 \in \lambda(v_3) \end{gathered}$$

By applying the definition of MD element instances (function λ in Definition 9.4):

$$\begin{gathered} a_1 \triangleright a_3 \Rightarrow \exists (a_1, (a_1, a_3), a_3) : \\ \exists \mu_1 = \{s_1 \mapsto a_1\} \in [\![\kappa(v_1)]\!] \wedge \\ \exists \mu_2 = \{s_1 \mapsto a_1, s_3 \mapsto a_3\} \in [\![\kappa((v_1, v_3))]\!] \wedge \\ \exists \mu_3 = \{s_3 \mapsto a_3\} \in [\![\kappa(v_3)]\!] \end{gathered}$$

Which, knowing what each $\kappa(v_i)$ is, can be written as:

$$\begin{gathered} a_1 \triangleright a_3 \Rightarrow \exists (a_1, (a_1, a_3), a_3) : \\ \exists \mu_1 = \{s_1 \mapsto a_1\} \in [\![\pi_{\{s_1\}}(p_1)]\!] \wedge \\ \exists \mu_2 = \{s_1 \mapsto a_1, s_3 \mapsto a_3\} \in [\![\pi_{\{s_1,s_3\}}(p_2)]\!] \wedge \\ \exists \mu_3 = \{s_3 \mapsto a_3\} \in [\![\pi_{\{s_3\}}(p_3)]\!] \end{gathered}$$

Let us take $\mu_1 \in [\![\pi_{\{s_1\}}(p_1)]\!]$ and $\mu_2 \in [\![\pi_{\{s_1,s_3\}}(p_2)]\!]$. Note that $\mu_1 \sim \mu_2$ as: (i) $dom(\mu_1) = \{s_1\}$, (ii) $dom(\mu_2) = \{s_1, s_3\}$, (iii) $dom(\mu_1) \cap dom(\mu_2) = \{s_1\}$, and (iv) μ_1 and μ_2 agree on s_1 with the value a_1. Note that $\mu_1 \cup \mu_2 = \mu_2 = \{s_1 \mapsto a_1, s_3 \mapsto a_3\}$. It follows that $\mu_2 \in [\![\pi_{\{s_1\}}(p_1)]\!] \bowtie [\![\pi_{\{s_1,s_3\}}(p_2)]\!]$ (the definition of join). Now let us take $\mu_2 = \{s_1 \mapsto a_1, s_3 \mapsto a_3\} \in [\![\pi_{\{s_1\}}(p_1)]\!] \bowtie [\![\pi_{\{s_1,s_3\}}(p_2)]\!]$ and $\mu_3 = \{s_3 \mapsto a_3\} \in [\![\pi_{\{s_3\}}(p_3)]\!]$. Note that $\mu_2 \sim \mu_3$ as: (i) $dom(\mu_2) = \{s_1, s_3\}$, (ii) $dom(\mu_3) = \{s_3\}$, (iii) $dom(\mu_2) \cap dom(\mu_3) = \{s_3\}$, and (iv) μ_2 and μ_3 agree on s_3 with the value a_3. note that $\mu_2 \cup \mu_3 = \mu_2 = \{s_1 \mapsto a_1, s_3 \mapsto a_3\}$. It follows that $\mu_2 \in ([\![\pi_{\{s_1\}}(p_1)]\!] \bowtie [\![\pi_{\{s_1,s_3\}}(p_2)]\!]) \bowtie [\![\pi_{\{s_3\}}(p_3)]\!]$ (definition of join). From the definition of projection and Proposition 3, it follows that:

$$\mu_2 = \{s_1 \mapsto a_1, s_3 \mapsto a_3\} \in \pi_{\{s_1,s_3\}} \Big([\![\pi_{\{s_1\}}(p_1)]\!] \bowtie [\![\pi_{\{s_1,s_3\}}(p_2)]\!] \bowtie [\![\pi_{\{s_3\}}(p_3)]\!] \Big)$$

which is the right-hand side (RIGHT) of Relation 2.

RIGHT ⇒ LEFT. The proof is done by starting form the right-hand side of Relation 1 and proceeding until we get the left-hand side. From the right-hand side of relation 1, and by applying the definition of projection, it follows that:

$$\forall a_1, \forall a_3 : \Bigg(\Big(\mu = \{s_1 \mapsto a_1, s_3 \mapsto a_3\} \in [\![\pi_{\{s_1,s_3\}} \left(\kappa(v_1) \bowtie \kappa(v_2) \bowtie \kappa(v_3) \right)]\!] \Big) \Rightarrow \\ \Big(\exists \mu' \in [\![\left(\kappa(v_1) \bowtie \kappa(v_2) \bowtie \kappa(v_3) \right)]\!] : \mu'_{|\{s_1,s_3\}} = \mu \Big) \Bigg)$$

Starting from the right-hand side of the previous relation, and by applying the definition of join, it follows that:

$$\begin{gathered} \exists \mu' \in [\![\left(\kappa(v_1) \bowtie \kappa(v_2) \bowtie \kappa(v_3) \right)]\!] : \mu'_{|\{s_1,s_3\}} = \mu \Rightarrow \\ \exists \mu'_1 = \{s_1 \mapsto a'_1\} \in [\![\kappa(v_1)]\!], \\ \exists \mu'_2 = \{s_1 \mapsto a''_1, s_3 \mapsto a''_3\} \in [\![\kappa(v_2)]\!], \\ \exists \mu'_3 = \{s_3 \mapsto a'''_3\} \in [\![\kappa(v_3)]\!] : \\ \mu' = ((\mu'_1 \cup \mu'_2) \cup \mu'_3) \wedge (\mu'_1 \sim \mu'_2) \wedge ((\mu'_1 \cup \mu'_2) \sim \mu'_3) \end{gathered} \tag{3}$$

From the right-hand side of the relation 3, consider: (i) $\mu'_1 = \{s_1 \mapsto a'_1\} \in [\![\kappa(v_1)]\!]$, (ii) $\mu'_2 = \{s_1 \mapsto a''_1, s_3 \mapsto a''_3\} \in [\![\kappa(v_2)]\!]$, and (iii) $\mu'_1 \sim \mu'_2$. From (i), (ii), and (iii), and as $\mu'_1 \sim \mu'_2$ means that they agree on $dom(\mu'_1) \cap dom(\mu'_2)$, it follows that $\mu'_1(s_1) = \mu'_2(s_1)$ and, hence, it follows that $a'_1 = a''_1$. From the right-hand side of the relation 3, consider (i) $\mu'_1 \cup \mu'_2 = \{s_1 \mapsto a'_1, s_3 \mapsto a''_3\}$, (ii) $\mu'_3 = \{s_3 \mapsto a'''_3\}$, and (iii) $(\mu'_1 \cup \mu'_2) \sim \mu'_3$. From (i), (ii), and (iii), and as $(\mu'_1 \cup \mu'_2) \sim \mu'_3$ means that they agree on $dom(\mu'_1 \cup \mu'_2) \cap dom(\mu'_3)$ it follows that $(\mu'_1 \cup \mu'_2)(s_3) = \mu'_3(s_3)$ and, hence, $a''_3 = a'''_3$. Hence, μ' amounts to $\mu' = \{s_1 \mapsto a'_1, s_3 \mapsto a'''_3\}$. We know that (i) $\mu = \{s_1 \mapsto a_1, s_3 \mapsto a_3\}$, (ii) $\mu' = \{s_1 \mapsto a'_1, s_3 \mapsto a'''_3\}$, and (iii) $\mu'_{|\{s_1,s_3\}} = \mu$, it follows from the definition of projection that $\mu = \mu'$ and, hence, $a_1 = a'_1$ and $a_3 = a'''_3$. From $\{s_1 \mapsto a'_1\} \in [\![\kappa(v_1)]\!]$ in relation 3 and $a_1 = a'_1$ it follows that (i) $a_1 \in \lambda(v_1)$. In the same way, it follows that (ii) $(a_1, a_3) \in \lambda(v_2)$, and (iii) $a_3 \in \lambda(v_3)$. Hence, from the definition of connected instances (Definition 9.5), (i), (ii), and (iii), it follows that $a_1 \triangleright a_3$ and, hence, $(a_1, a_3) \in A_3$ which is the left-hand side of Relation 1.

Induction. We will prove that if the relation of the proposition is correct for m, then it is correct for $m + 2$, i.e., $A_m = B_m \Rightarrow A_{m+2} = B_{m+2}$. The proof follows by double-inclusion. On the one hand, correctness for m is expressed as:

$$\forall a_1, \forall a_m : \Bigg(\Big(a_1 \ \triangleright \ a_m \Big) \Leftrightarrow \Big(\exists \mu = \{s_1 \mapsto a_1, s_m \mapsto a_m\} \in [\![\pi_{\{s_1,s_m\}} (\bigwedge_{i=1}^{m} \kappa(v_i))]\!] \Big) \Bigg) \tag{4}$$

On the other hand, correctness for $m + 2$ is expressed as:

$$\forall a_1, \forall a_{m+2} : \Bigg(\Big(a_1 \ \triangleright \ a_{m+2} \Big) \Leftrightarrow \Big(\exists \mu = \{s_1 \mapsto a_1, s_{m+2} \mapsto a_{m+2}\} \in [\![\pi_{\{s_1,s_{m+2}\}} (\bigwedge_{i=1}^{m+2} \kappa(v_i))]\!] \Big) \Bigg) \tag{5}$$

LEFT $\Rightarrow$ RIGHT. The proof is done by starting form the left-hand side of Relation 5 and proceeding until we get the right-hand side. From the definition of connectivity (Definition 9.5), it follows that:

$$a_1 \triangleright a_{m+2} \Rightarrow \exists a_m \in \lambda(v_m) : a_1 \triangleright a_m \wedge a_m \triangleright a_{m+2} \tag{6}$$

From the right-hand side of the relation 6, and as we assumed the correctness for m (relation 4), it follows that:

$$a_1 \ \triangleright \ a_m \Rightarrow \exists \mu_1 = \{s_1 \mapsto a_1, s_m \mapsto a_m\} \in [\![\pi_{\{s_1,s_m\}} \left(\bigwedge_{i=1}^{m} \kappa(v_i) \right)]\!] \tag{7}$$

From the right-hand side of the relation 6 and as we know that the proposition is correct for $m = 3$, we get:

$$a_m \ \triangleright \ a_{m+2} \Rightarrow \exists \mu_2 = \{s_m \mapsto a_m, s_{m+2} \mapsto a_{m+2}\} \in [\![\pi_{\{s_m,s_{m+2}\}} \left(\bigwedge_{i=m}^{m+2} \kappa(v_i) \right)]\!] \tag{8}$$

Let us take the right-hand sides of the relations 7 and 8, and apply Proposition 9, we get the following relation

$$\exists \mu' = \{s_1 \mapsto a_1, s_{m+2} \mapsto a_{m+2}\} \in [\![\pi_{\{s_1,s_{m+2}\}} \left(\bigwedge_{i=1}^{m+2} \kappa(v_i) \right)]\!] \tag{9}$$

Which is the right-hand side of Relation 5.

RIGHT $\Rightarrow$ LEFT. The proof is done by starting form the right-hand side of Relation 5 and proceeding until we reach the left-hand side. By applying Proposition 9 to the right hand side of the relation 5, it follows that:

$$\mu = \{s_1 \mapsto a_1, s_{m+2} \mapsto a_{m+2}\} \in [\![\pi_{\{s_1,s_{m+2}\}} \left(\bigwedge_{i=1}^{m+2} \kappa(v_i) \right)]\!] \Rightarrow \exists a_m : \tag{10}$$

$$\{s_1 \mapsto a_1, s_m \mapsto a_m\} \in [\![\pi_{\{s_1,s_m\}} \left(\bigwedge_{i=1}^{m} \kappa(v_i) \right)]\!] \wedge \tag{11}$$

$$\{s_m \mapsto a_m, s_{m+2} \mapsto a_{m+2}\} \in [\![\pi_{\{s_m,s_{m+2}\}} \left(\bigwedge_{i=m}^{m+2} \kappa(v_i) \right)]\!] \tag{12}$$

We supposed that the proposition is correct for m. Hence, from the relation 11, it follows that (i) $a_1 \triangleright a_m$. We know that the proposition is correct for $m = 3$. Hence from the relation 12, it follows that (ii) $a_m \triangleright a_{m+2}$. From (i), (ii), and the definition of connectedness (Definition 9.5), it follows that $a_1 \triangleright a_{m+2}$, which is the left-hand side of Relation 5. □

2.2.2 SPARQL Query of a Base Set of Grouping (Definition 9.13)

Given a group-by list $(l_1, ..., l_n)$. Let X be the respective base set of grouping. Furthermore, let the set X' be defined as $X' = \{(b_f, b_1, ..., b_n) \in \mathcal{T}^{n+1} \mid \exists \mu = \{s_f \mapsto b_f, s_{1,m_1} \mapsto b_1, ..., s_{n,m_n} \mapsto b_n\} \in [\![q_X]\!] : q_X = \alpha\left(\pi_{\{s_f, s_{1,m_1}, ..., s_{n,m_n}\}}\left(\bigwedge_{i=1}^{n}\left(\pi_{\{s_f, s_{i,m_i}\}}\left(\bigwedge_{j=1}^{m_i}\kappa(v_{i,j})\right)\right)\right)\right)\}$. Then the following equality holds: $X = X'$.

Proof. Note that we prove that each solution mapping in the evaluation of q_X results in a respective member in X, and that each member in X has a respective solution mapping in the evaluation of q_X. Note that duplication elimination operator α can be safely dropped form the query q_X since X' is a set, i.e., free of duplicates.

The proof is done by double inclusion and induction. Whereas the following proof is for the case where $n \geq 2$, the proof is intuitive when $n = 1$ which amounts to connectivity as in Proof 2.2.1, as we show in the following. In particular, the query becomes:

$$\alpha\left(\pi_{\{s_f, s_{1,m_1}\}}\left(\pi_{\{s_f, s_{1,m_1}\}}\left(\bigwedge_{j=1}^{m_1}\kappa(v_{1,j})\right)\right)\right)$$

Which, according to Proposition 7, can be written as:

$$\alpha\left(\pi_{\{s_f, s_{1,m_1}\}}\left(\bigwedge_{j=1}^{m_1}\kappa(v_{1,j})\right)\right)$$

The proof proceeds as following. We first prove that the relation is correct for $n = 2$ by double-inclusion. Afterwards, we proceed with induction and show, using double-inclusion again, that if the relation is correct for n, then it is correct for $n + 1$. Let X_n and X'_n where $n \in \mathbb{N}$ denote the sets X and X' when the length of the considered group-by list $(l_1, ..., l_n)$ is n. We prove the equality without

Correctness for $n = 2$. Before we proceed with the proof, we make the following simplification: Let s_i denote s_{i,m_i}, e.g., s_2 denotes s_{2,m_2}, which can be assumed without ambiguity. Correctness

for $n = 2$ means that: $\forall a_f, \forall a_1, \forall a_2$, the following equivalence holds:

$$\Big((a_f, a_1, a_2) \in X_2\Big) \Leftrightarrow$$

$$\left(\begin{array}{l} \exists \mu = \{s_f \mapsto a_f, s_1 \mapsto a_1, s_2 \mapsto a_2\} \in \\ \\ [\![\alpha \left(\pi_{\{s_f,s_1,s_2\}} \left(\left(\pi_{\{s_f,s_1\}} \Big(\bigwedge_{j=1}^{m_1} q_{1,j} \Big) \right) \bowtie \left(\pi_{\{s_f,s_2\}} \Big(\bigwedge_{j=1}^{m_2} q_{2,j} \Big) \right) \right) \right)]\!] \end{array}\right)$$

Which can be simplified, by removing the duplicate removal operator α:

$$\Big((a_f, a_1, a_2) \in X_2\Big) \Leftrightarrow$$

$$\left(\begin{array}{l} \exists \mu = \{s_f \mapsto a_f, s_1 \mapsto a_1, s_2 \mapsto a_2\} \in \\ \\ [\![\underbrace{\left(\underbrace{\left(\pi_{\{s_f,s_1\}} \Big(\bigwedge_{j=1}^{m_1} q_{1,j} \Big) \right)}_{R_1} \bowtie \underbrace{\left(\pi_{\{s_f,s_2\}} \Big(\bigwedge_{j=1}^{m_2} q_{2,j} \Big) \right)}_{R_2} \right)}_{R}]\!] \end{array}\right)$$

We first make some shortcuts. Let $R_1 \coloneqq \pi_{\{s_f,s_1\}} \left(\bigwedge_{j=1}^{m_1} q_{1,j} \right)$. Let $R_2 \coloneqq \pi_{\{s_f,s_2\}} \left(\bigwedge_{j=1}^{m_2} q_{2,j} \right)$. Let $R \coloneqq \left(\pi_{\{s_f,s_1\}} \Big(\bigwedge_{j=1}^{m_1} q_{1,j} \Big) \right) \bowtie \left(\pi_{\{s_f,s_2\}} \Big(\bigwedge_{j=1}^{m_2} q_{2,j} \Big) \right)$. As $vars(R_1 \bowtie R_2) = vars(R_1) \cup vars(R_2) = \{s_f, s_1, s_2\}$ [2], [4](refer to 1.1), the most outer projection can be eliminated (Proposition 7),

and the required proof amounts to: $\forall a_f, \forall a_1, \forall a_2$, the following equivalence holds:

$$\Big((a_f, a_1, a_2) \in X_2\Big) \Leftrightarrow \left(\begin{array}{l} \exists \mu = \{s_f \mapsto a_f, s_1 \mapsto a_1, s_2 \mapsto a_2\} \in \\ \\ \underbrace{[\![\underbrace{\left(\pi_{\{s_f, s_{m_1}\}} \Big(\bigwedge_{j=1}^{m_1} q_{1,j} \Big) \right)}_{R_1} \bowtie \underbrace{\left(\pi_{\{s_f, s_{m_2}\}} \Big(\bigwedge_{j=1}^{m_2} q_{2,j} \Big) \right)}_{R_2}]\!]}_{R} \end{array} \right) \tag{13}$$

LEFT ⇒ RIGHT. The proof follows by starting from the left-hand side of Relation 13 and proceeding until we get the right-hand side. We know from the definition of base set of grouping (Definition 9.13) that $(a_f, a_1, a_2) \in X_2 \Rightarrow a_f \triangleright a_1 \wedge a_f \triangleright a_2$. We know also from SPARQL of MD path (Section 2.2.1) that $a_f \triangleright a_1 \Rightarrow \exists \mu_1 = \{s_f \mapsto a_f, s_1 \mapsto a_1\} \in [\![\pi_{\{s_f, s_1\}}(\bigwedge_{j=1}^{m_1} q_{1,j})]\!]$, and that $a_f \triangleright a_2 \Rightarrow \exists \mu_2 = \{s_f \mapsto a_f, s_2 \mapsto a_2\} \in [\![\pi_{\{s_f, s_2\}}(\bigwedge_{j=1}^{m_2} q_{2,j})]\!]$. By comparison with the definition of each of R_1 and R_2 it follows that $\mu_1 \in [\![R_1]\!]$ and that $\mu_2 \in [\![R_2]\!]$. Note that $\mu_1 \sim \mu_2$ and, hence, $\mu_1 \cup \mu_2 \in [\![R_1 \bowtie R_2]\!]$ (join definition). As $\mu_1 \cup \mu_2 = \{s_f \mapsto a_f, s_1 \mapsto a_1, s_2 \mapsto a_2\}$, it follows that $\{s_f \mapsto a_f, s_1 \mapsto a_1, s_2 \mapsto a_2\} \in [\![R_1 \bowtie R_2]\!]$ which is the right-hand side of Relation 13.

RIGHT ⇒ LEFT. The proof follows by starting from the right-hand side of Relation 13 and proceeding until we get the left-hand side. From the right-hand side of relation 13 we know that $\mu = \{s_f \mapsto a_f, s_1 \mapsto a_1, s_2 \mapsto a_2\} \in [\![R_1 \bowtie R_2]\!]$, which means that (i) $\exists \mu_1 \in [\![R_1]\!]$, (ii) $\exists \mu_2 \in [\![R_2]\!]$, (iii) $\mu_1 \sim \mu_2$, and (iv) $\mu = \mu_1 \cup \mu_2$. We know that μ_1 is of the form $\{s_f \mapsto a'_f, s_1 \mapsto a'_1\}$ and that μ_2 is of the form $\{s_f \mapsto a''_f, s_2 \mapsto a''_2\}$ and that $a'_f = a''_f$ as $\mu_1 \sim \mu_2$ and $dom(\mu_1) \cap dom(\mu_2) = \{s_f\}$. Hence, by applying the forms of μ_1 and μ_2 in $\mu = \mu_1 \cup \mu_2$, it results that $\mu = \{s_f \mapsto a'_f, s_1 \mapsto a'_1, s_2 \mapsto a''_2\}$, by comparing to $\mu = \{s_f \mapsto a_f, s_1 \mapsto a_1, s_2 \mapsto a_2\}$, it follows that: $a_f = a'_f$, $a_1 = a'_1$, and $a_2 = a''_2$ and hence: $\mu_1 = \{s_f \mapsto a_f, s_1 \mapsto a_1\}$ and $\mu_2 = \{s_f \mapsto a_f, s_2 \mapsto a_2\}$. By comparing the new forms for μ_1 and μ_2 with SPARQL query of MD path (Section 2.2.1), it follows that $a_f \triangleright a_1$ and that $a_f \triangleright a_2$, and by comparison with the definition of the base set of grouping (Definition 9.12), it follows that $(a_f, a_1, a_2) \in X_2$.

Induction. Now we are going to prove that if the relation is correct for n, then it is correct for $n+1$. Correctness for n means that $\forall a_f, \forall a_1, ..., \forall a_n$, the following equivalence holds:

$$\Big((a_f, a_1, ..., a_n) \in X_n \Leftrightarrow \Big)$$

$$\left(\exists \mu = \{s_f \mapsto a_f, s_1 \mapsto a_1, ..., s_n \mapsto a_n\} \in \right.$$

$$\left. [\![\alpha \left(\pi_{\{s_f, s_1, ..., s_n\}} \left(\left(\pi_{\{s_f, s_1\}} \Big(\bigwedge_{j=1}^{m_1} q_{1,j} \Big) \right) \bowtie ... \bowtie \left(\pi_{\{s_f, s_n\}} \Big(\bigwedge_{j=1}^{m_n} q_{n,j} \Big) \right) \right) \right)]\!] \right)$$

The previous equivalence can be simplified to by dropping duplicate removal and outermost projection (Proposition 7, similarly to the reasoning behind relation 13):

$$\Big((a_f, a_1, ..., a_n) \in X_n\Big) \Leftrightarrow$$

$$\left(\exists \mu = \{s_f \mapsto a_f, s_1 \mapsto a_1, ..., s_n \mapsto a_n\} \in \right.$$

$$\left. [\![\left(\pi_{\{s_f, s_1\}} \Big(\bigwedge_{j=1}^{m_1} q_{1,j} \Big) \right) \bowtie ... \bowtie \left(\pi_{\{s_f, s_n\}} \Big(\bigwedge_{j=1}^{m_n} q_{n,j} \Big) \right)]\!] \right)$$

Similarly, correctness for $n+1$ means that $\forall a_f, \forall a_1, ..., \forall a_n, \forall a_{n+1}$, the following equivalence holds:

$$\Big((a_f, a_1, ..., a_n, a_{n+1}) \in X_{n+1}\Big) \Leftrightarrow$$

$$\left(\exists \mu = \{s_f \mapsto a_f, s_1 \mapsto a_1, ..., s_n \mapsto a_n, s_{n+1} \mapsto a_{n+1}\} \in \right.$$

$$\left. \underbrace{[\![\underbrace{\left(\pi_{\{s_f, s_{m_1}\}} \Big(\bigwedge_{j=1}^{m_1} q_{1,j} \Big) \right) \bowtie ... \bowtie \left(\pi_{\{s_f, s_{m_n}\}} \Big(\bigwedge_{j=1}^{m_n} q_{n,j} \Big) \right)}_{R_{1..n}} \bowtie \underbrace{\left(\pi_{\{s_f, s_{m_{n+1}}\}} \Big(\bigwedge_{j=1}^{m_{n+1}} q_{n+1,j} \Big) \right)}_{R_{n+1}}]\!]}_{R} \right) \tag{14}$$

We first make some shortcuts. Let $R_{1..n} \coloneqq \left(\pi_{\{s_f, s_1\}} \Big(\bigwedge_{j=1}^{m_1} q_{1,j} \Big) \right) \bowtie ... \bowtie \left(\pi_{\{s_f, s_n\}} \Big(\bigwedge_{j=1}^{m_n} q_{n,j} \Big) \right)$, let $R_{n+1} \coloneqq \left(\pi_{\{s_f, s_{n+1}\}} \Big(\bigwedge_{j=1}^{m_{n+1}} q_{n+1,j} \Big) \right)$, and finally let $R \coloneqq R_1 \bowtie R_2$. From the definition of the

base set of grouping (Definition 9.12), it follows that:

$$(a_f, a_1, ..., a_n, a_{n+1}) \in X_{n+1} \Leftrightarrow (a_f, a_1, ..., a_n) \in X_n \ \wedge \ a_f \triangleright a_{n+1} \ \wedge a_{n+1} \in \lambda(l_{n+1}) \tag{15}$$

Furthermore, the joins in $R_{1..n}$ can be collapsed into a big join which allows to write R as follows:

$$\underbrace{[\![\bigwedge_{i=1}^{n} \underbrace{\left(\pi_{\{s_f, s_{m_i}\}} \left(\bigwedge_{j=1}^{m_i} q_{i,j}\right)\right)}_{R_{1..n}} \bowtie \underbrace{\left(\pi_{\{s_f, s_{m_{n+1}}\}} \left(\bigwedge_{j=1}^{m_{n+1}} q_{n+1,j}\right)\right)}_{R_{n+1}}]\!]}_{R} \tag{16}$$

From relations 15 and 16, relation 14 can be rewritten to the following form which will be used in the following proof.

$$\begin{aligned} &\big((a_f, a_1, ..., a_n) \in X_n \ \wedge \ a_f \triangleright a_{n+1} \ \wedge \ a_{n+1} \in \lambda(l_{n+1})\big) \Leftrightarrow \\ &\Bigg(\exists \mu = \{s_f \mapsto a_f, s_1 \mapsto a_1, ..., s_n \mapsto a_n, s_{n+1} \mapsto a_{n+1}\} \in \\ &\underbrace{[\![\bigwedge_{i=1}^{n} \underbrace{\left(\pi_{\{s_f, s_{m_i}\}} \left(\bigwedge_{j=1}^{m_i} q_{i,j}\right)\right)}_{R_{1..n}} \bowtie \underbrace{\left(\pi_{\{s_f, s_{m_{n+1}}\}} \left(\bigwedge_{j=1}^{m_{n+1}} q_{n+1,j}\right)\right)}_{R_{n+1}}]\!]}_{R} \Bigg) \end{aligned} \tag{17}$$

LEFT ⇒ RIGHT. The proof is conducted by starting from the left-hand side of Relation 17 and proceeding until we get the right-hand side. We know from the correctness of the relation for n that $(a_f, ..., a_n) \in X_n$ implies that (i) $\mu_{1..n} = \{s_f \mapsto a_f, s_1 \mapsto a_1, ..., s_n \mapsto a_n\} \in [\![R_{1..n}]\!]$. From SPARQL query of an MD path (Section 2.2.1), it follows that $a_f \triangleright a_{n+1}$ implies that (ii) $\mu_{n+1} = \{s_f \mapsto a_f, s_{n+1} \mapsto a_{n+1}\} \in [\![R_{n+1}]\!]$. From (i) and (ii), and the definition of join, it follows that $\mu_{1..n} \cup \mu_{n+1} = \{s_f \mapsto a_f, s_1 \mapsto a_1, ..., s_n \mapsto a_n, s_{n+1} \mapsto a_{n+1}\} \in [\![R_{1..n} \bowtie R_{n+1}]\!]$ which is the right-hand side of Relation 17.

RIGHT ⇒ LEFT. The proof is conducted by starting from the right-hand side of Relation 17 and proceeding until we get the left-hand side. From the definition of join, we know that $\mu = \{s_f \mapsto a_f, s_1 \mapsto a_1, ..., s_n \mapsto a_n, s_{n+1} \mapsto a_{n+1}\} \in [\![R]\!] \Rightarrow \exists \mu_{1..n} = \{s_f \mapsto a'_f, s_1 \mapsto a'_1, ..., s_n \mapsto a'_n\} \in [\![R_{1..n}]\!]$, $\exists \mu_{n+1} = \{s_f \mapsto a''_f, s_{n+1} \mapsto a''_{n+1}\} \in [\![R_{n+1}]\!]$: $\mu_{1..n} \sim \mu_{n+1}$. We know that $\mu_{1..n} \sim \mu_{n+1}$ means that $\mu_{1..n}(s_f) = \mu_{n+1}(s_f)$, and, hence, $a'_f = a''_f$. As a consequence, it follows that $\mu = \mu_{1..n} \cup \mu_{n+1} = \{s_f \mapsto a'_f, s_1 \mapsto a'_1, ..., s_n \mapsto a'_n, s_{n+1} \mapsto a''_{n+1}\}$. By comparison with $\mu = \{s_f \mapsto a_f, s_1 \mapsto a_1, ..., s_n \mapsto a_n, s_{n+1} \mapsto a_{n+1}\}$ it follows that $a_f = a'_f$, $a_1 = a'_1$, ...,

$a_n = a'_n$, and $a_{n+1} = a''_{n+1}$ and, hence, $\mu_{1..n} = \{s_f \mapsto a_f, s_1 \mapsto a_1, ..., s_n \mapsto a_n\}$ and $\mu_{n+1} = \{s_f \mapsto a_f, s_{n+1} \mapsto a_{n+1}\}$. As $\mu_{n+1} \in [\![R_{n+1}]\!]$ and $\mu_{n+1} = \{s_f \mapsto a_f, s_{n+1} \mapsto a_{n+1}\}$, it follows from SPARQL query of an MD path (Section 2.2.1) that (i) $a_f \triangleright a_{n+1}$ and that $a_{n+1} \in \lambda(l_{n+1})$. As the relation is correct for n, and $\mu_{1..n} = \{s_f \mapsto a_f, s_1 \mapsto a_1, ..., s_n \mapsto a_n\} \in [\![R_{1..n}]\!]$ it follows that (ii) $(a_f, a_1, ..., a_n) \in X_n$. From (i) and (ii) we get the left-hand side of Relation 17. □

2.2.3 SPARQL Query of a Base Set of Measure (Definition 9.15)

Given a measure $msr \in M$, Let the set $Y = \{(a_f, a_{msr}) \in \lambda((f, msr)) \mid f \in \mathscr{F}\}$ be the base set of the measure msr. Let $Y' = \{(b_f, b_{msr}) \in \mathcal{T} \times \mathcal{L} \mid \exists \mu = \{s_f \mapsto b_f, s_{msr} \mapsto b_{msr}\} \in [\![q_Y]\!]$ with $q_Y = \alpha\left(\pi_{\{s_f, s_{msr}\}}\left(\kappa(f) \bowtie \kappa((f, msr))\right)\right)\}$. Then the following equality holds: $Y = Y'$.

Proof. Note that we prove that each solution mapping in the evaluation of q_Y results in a respective member in Y, and that each member in Y has a respective solution mapping in the evaluation of q_Y. Note that duplication elimination operator α can be safely dropped form the query q_Y since Y' is a set, i.e., free of duplicates.

The proof is done by double-inclusion. We need to prove that $\forall a_f, \forall a_{msr}$, the following equivalence holds:

$$\Big((a_f, a_{msr}) \in Y\Big) \Leftrightarrow$$
$$\left(\exists \mu = \{s_f \mapsto a_f, s_{msr} \mapsto a_{msr}\} \in [\![\alpha\left(\pi_{\{s_f, s_{msr}\}}\left(\kappa(f) \bowtie \kappa((f, msr))\right)\right)]\!]\right)$$

Which can be simplified by removing the duplicate removal operator α as follows:

$$\Big((a_f, a_{msr}) \in Y\Big) \Leftrightarrow$$
$$\left(\exists \mu = \{s_f \mapsto a_f, s_{msr} \mapsto a_{msr}\} \in [\![\underbrace{\pi_{\{s_f, s_{msr}\}}\left(\kappa(f) \bowtie \kappa((f, msr))\right)}_{R}]\!]\right) \tag{18}$$

We first make the shortcut: $R \coloneqq \pi_{\{s_f, s_{msr}\}}\left(\kappa(f) \bowtie \kappa((f, msr))\right)$. As $vars(R) = vars(\kappa(f)) \cup vars(\kappa((f, msr))) = \{s_f, s_{msr}\}$ (refer to 1.1), the outermost projection in R can be eliminated (Proposition 7), and the previous equivalence 18 can be further simplified to:

$$\Big((a_f, a_{msr}) \in Y\Big) \Leftrightarrow$$
$$\left(\exists \mu = \{s_f \mapsto a_f, s_{msr} \mapsto a_{msr}\} \in [\![\kappa(f) \bowtie \kappa((f, msr))]\!]\right) \tag{19}$$

We will use the simplified Relation 19 for the proof in the following.

LEFT $\Rightarrow$ RIGHT. The proof is done by starting from the left-hand side of Relation 19 and proceeding until we get the right-hand side. We know from Definition 9.14 that $(a_f, a_{msr}) \in Y$ implies that (i) $a_f \in \lambda(f)$, and (ii) $(a_f, a_{msr}) \in \lambda((f, msr))$. From (i) and Definition 9.4, it follows that (i′) $\exists \mu_1 = \{s_f \mapsto a_f\} \in [\![\kappa(f)]\!]$. From (ii) and Definition 9.4, it follows that (ii′) $\exists \mu_2 = \{s_f \mapsto a_f, s_{msr} \mapsto a_{msr}\} \in [\![\kappa((f, msr))]\!]$. From (i′), (ii′), and the join definition, it follows that $\mu_1 \cup \mu_2 = \{s_f \mapsto a_f, s_{msr} \mapsto a_{msr}\} \in [\![\kappa(f) \bowtie \kappa((f, msr))]\!]$, which is the right-hand side of Relation 19.

RIGHT $\Rightarrow$ LEFT. The proof is done by starting from the right-hand side of Relation 19 and proceeding until we get the left-hand side. Starting from the right-hand side of Relation 19, and

according to the definition of join, it follows that: (i) $\exists \mu' = \{s_f \mapsto a'_f\} \in [\![\kappa(f)]\!]$, (ii) $\exists \mu'' = \{s_f \mapsto a''_f, s_{msr} \mapsto a''_{msr}\} \in [\![\kappa((f, msr))]\!]$, (iii) $\mu' \sim \mu''$, and (iv) $\mu' \cup \mu'' = \mu$. From (iii), it follows that $a'_f = a''_f$. Hence, $\mu'' = \{s_f \mapsto a'_f, s_{msr} \mapsto a''_{msr}\}$. If we apply the new definition of μ'' in (iv), it follows that $\mu = \mu' \cup \mu'' = \{s_f \mapsto a'_f\} \cup \{s_f \mapsto a'_f, s_{msr} \mapsto a''_{msr}\} = \{s_f \mapsto a'_f, s_{msr} \mapsto a''_{msr}\}$ which implies that $a_f = a'_f$ and that $a_{msr} = a''_{msr}$. As $a'_f \in \lambda(f)$ (i), and $(a''_f, a''_{msr}) \in \lambda((f, msr))$ (ii), it follows that $a_f \in \lambda(f)$ and $(a_f, a_{msr}) \in \lambda(f, msr)$ which, when applied in Definition 9.14, leads to: $(a_f, a_{msr}) \in Y$, which is the left-hand side of Relation 19. □

2.2.4 SPARQL Pattern of a Base Set of Aggregation (Definition 9.17)

Given: (i) $(l_1, ..., l_n)$ a group-by list where its base set of grouping is X and its query is q_X, (ii) $msr \in M$ a measure where its base set of measure is Y and the respective query is q_Y, and (iii) Z is the respective base set of aggregation formed from X and Y. Let the multiset Z' be defined as $Z' = \{\!\{(a_f, a_{msr}, a_1, ..., a_n) \in \mathcal{T}^{n+2} \mid \exists \mu = \{s_f \mapsto a_f, s_{msr} \mapsto a_{msr}, s_1 \mapsto a_1, ..., s_n \mapsto a_n\} \in [\![q_Z]\!]_G\}\!\}$, where q_Z is defined as:

$$\underbrace{\underbrace{\alpha\Big(\pi_{\{s_f, s_{msr}\}}\big(\kappa(f) \bowtie \kappa((f, msr))\big)\Big)}_{q_X} \bowtie \underbrace{\alpha\left(\pi_{\{s_f, s_{1,m_1}, ..., s_{n,m_n}\}}\left(\bigwedge_{i=1}^{n}\left(\pi_{\{s_f, s_{i,m_i}\}}\Big(\bigwedge_{j=1}^{m_i}\kappa(v_{i,j})\Big)\right)\right)\right)}_{q_Y}}_{q_Z}$$

where the query pattern joins the query q_X of the base set of grouping (Definition 9.13) with the query q_Y of the base set of measure (Definition 9.14). Then the following equivalence holds: $Z \equiv Z'$. We define the equivalence of a set and a multiset as follows: Given a set S and a multiset $M = (\Omega, card)$, we write $S \equiv M$ if and only if $S = \Omega \ \wedge \ \forall x \in \Omega : card(x) = 1$.

Proof. The proof is done by double inclusion and afterwards showing that the cardinality of each item in Z' is 1, i.e., Z and Z' are equivalent.

$\boldsymbol{x \in Z \Rightarrow x \in Z'}$**.** From the definition of Z (Definition 9.16), we know that $\forall (a_f, a_{msr}, a_{1,m_1}, ..., a_{n,m_n}) \in Z : (a_f, a_{1,m_1}, ..., a_{n,m_n}) \in X \wedge (a_f, a_{msr}) \in Y$. From Section 2.2.2 and Section 2.2.3, we know that (i) $\mu_X = \{s_f \mapsto a_f, s_{1,m_1} \mapsto a_{1,m_1}, ..., s_{n,m_n} \mapsto a_{n,m_n}\} \in [\![q_X]\!]$, and that (ii) $\mu_Y = \{s_f \mapsto a_f, s_{msr} \mapsto a_{msr}\} \in [\![q_Y]\!]$. From (i) and (ii), and according to the definition of join, it follows that $\mu_X \cup \mu_Y = \{s_f \mapsto a_f, s_{msr} \mapsto a_{msr}, s_{1,m_1} \mapsto a_{1,m_1}, ..., s_{n,m_n} \mapsto a_{n,m_n}\} \in [\![q_X \bowtie q_Y]\!] = [\![q_Z]\!]$, which when applied in the definition of Z' it implies that $(a_f, a_{msr}, a_{1,m_1}, ..., a_{n,m_n}) \in Z'$.

$\boldsymbol{x \in Z' \Rightarrow x \in Z}$**.** From the definition of q_Z, we know that $\forall \mu_Z = \{s_f \mapsto a_f, s_{msr} \mapsto a_{msr}, s_{1,m_1} \mapsto a_{1,m_1}, ..., \ s_{n,m_n} \mapsto a_{n,m_n}\} \in [\![q_Z]\!]$: $\exists \mu_X = \{s_f \mapsto a_f, s_{1,m_1} \mapsto a_{1,m_1}, ..., s_{n,m_n} \mapsto a_{n,m_n}\} \in [\![q_X]\!]$ and that $\exists \mu_Y = \{s_f \mapsto a_f, s_{msr} \mapsto a_{msr}\} \in [\![q_Y]\!]$ such that $\mu_X \sim \mu_Y$ and $\mu_Z = \mu_X \cup \mu_Y$. Hence, from Section 2.2.2 and Section 2.2.3, it follows that (i) $(a_f, a_{1,m_1}, ..., a_{n,m_n}) \in X$, and that (ii) $(a_f, a_{msr}) \in Y$. From (i), (ii), and the definition of Z (Definition 9.16), it follows that $(a_f, a_{msr}, a_{1,m_1}, ..., a_{n,m_n}) \in Z$.

Cardinality. We still need to show that the cardinality of each element in Z' is 1 which is true if the evaluation $[\![q_Z]\!]$ of the query q_Z does not result in any duplicates. Let us suppose that $\exists \mu_Z \in [\![q_Z]\!]$ with a cardinality more than 1, i.e., μ_Z is duplicated. According to the definitoin of join, the cardinality of μ_Z is the sum over the cardinalities of the participating $\mu_X \in [\![q_X]\!]$ and $\mu_Y \in [\![q_Y]\!]$ (where $\mu_X \sim \mu_Y$ and $\mu_Z = \mu_X \cup \mu_Y$). Let $M_Z = (\Omega_Z, card_Z) = [\![q_Z]\!]$, $M_X = (\Omega_X, card_X) = [\![q_X]\!]$, and $M_Y = (\Omega_Y, card_Y) = [\![q_Y]\!]$. In order for $card(\mu_Z)$ to be more than 1, there should be either one of two cases. CASE (1): A participating μ_X or μ_Y has a

cardinality more than 1 which is not possible, as the duplicate removal is applied in both q_X and q_Z. CASE (2): There exist more than one way to compose μ_Z, i.e., (i) $(\exists \mu'_X, \mu_X \in [\![q_X]\!]) \wedge (\exists \mu'_Y, \mu_Y \in [\![q_Y]\!]) \wedge (\mu'_X \neq \mu_X \vee \mu'_Y \neq \mu_Y) : (\mu'_X \sim \mu'_Y \wedge \mu_Z = \mu'_X \cup \mu'_Y) \ \wedge \ (\mu_X \sim \mu_Y \wedge \mu_Z = \mu_X \cup \mu_Y)$. (i) We know that $\mu_Z(s_{msr})$ and $\mu_Z(s_f)$ come from a solution mapping form $[\![q_Y]\!]$. If $\mu'_Y \neq \mu_Y$, then either $\mu_Y(s_{msr}) \neq \mu'_Y(s_{msr})$ or $\mu_Y(s_f) \neq \mu'_Y(s_f)$ which leads to contradiction as it will result in two different values of μ_Z. (ii) Similarly we know that $\mu_Z(s_f), ..., \mu_Z(s_{n,m_n})$ come from a solution mapping form $[\![q_X]\!]$. If $\mu'_X \neq \mu_X$, then either $\mu_X(s_f) \neq \mu'_X(s_f)$ or ... or $\mu_X(s_{n,m_n}) \neq \mu'_X(s_{n,m_n})$ which leads to contradiction as it will result in two different values of μ_Z. From (i) and (ii), it follows that our supposition is wrong. □

2.2.5 Final SPARQL Aggregation Query (Definition 9.18)

In the following, we show how the final aggregation query specified in Definition 9.18 (up to point right before the aggregation function `f` is applied) is equivalent to our definition of aggregation (Definition 9.9).

Proof. We first apply the definition of *grouping* [1], [3]. The result of grouping applied as per Definition 9.18 would be, for each coordinate $(a_{1,m_1}, ..., a_{n,m_n})$, a respective multiset of solution mappings as follows:

$$\begin{gathered} Group((s_{1,m_1}, ..., s_{n,m_n}), [\![q_Z]\!])(a_{1,m_1}, ..., a_{n,m_n}) = \\ \{\!\{\{s_f \mapsto a'_f, s_{msr} \mapsto a'_{msr}, s_{1,m_1} \mapsto a'_{1,m_1}, ..., s_{n,m_n} \mapsto a'_{n,m_n}\} \mid \\ \{s_f \mapsto a'_f,\ s_{msr} \mapsto a'_{msr},\ s_{1,m_1} \mapsto a'_{1,m_1},\ ...,\ s'_{n,m_n} \mapsto a_{n,m_n}\} \in [\![q_Z]\!] \\ \wedge\ a_{1,m_1} = a'_{1,m_1}\ \wedge\ ...\ \wedge\ a_{n,m_n} = a'_{n,m_n}\}\!\} \end{gathered} \tag{20}$$

Then, we apply the definition of aggregation [1], [3]. Each coordinate $(a_{1,m_1}, ..., a_{n,m_n})$ will have a respective multiset of values of the measure msr, which will be delivered to the aggregation function `f`, defined as follows[3]:

$$\begin{gathered} Aggregation((s_{msr}), \mathtt{f}, \{\}, Group((s_{1,m_1}, ..., s_{n,m_n}), [\![q_Z]\!]))(a_{1,m_1}, ..., a_{n,m_n}) = \\ ((a_{1,m_1}, ..., a_{n,m_n}), \mathtt{f}(\{\!\{a'_{msr} \mid \{s_f \mapsto a'_f, s_{msr} \mapsto a'_{msr}, s_{1,m_1} \mapsto a'_{1,m_1}, ..., s_{n,m_n} \mapsto a'_{n,m_n}\} \\ \in Group((s_{1,m_1}, ..., s_{n,m_n}), [\![q_Z]\!])(a_{1,m_1}, ..., a_{n,m_n})\}\!\})) \end{gathered} \tag{21}$$

We call the multiset that is used as input for function `f` for the coordinate $(a_{1,m_1}, ..., a_{n,m_n})$ as $multiset(a_{1,m_1}, ..., a_{n,m_n})$. Hence, we can write:

$$\begin{gathered} multiset(a_{1,m_1}, ..., a_{n,m_n}) = \\ \{\!\{a'_{msr} \mid \{s_f \mapsto a'_f, s_{msr} \mapsto a'_{msr}, s_{1,m_1} \mapsto a'_{1,m_1}, ..., s_{n,m_n} \mapsto a'_{n,m_n}\} \\ \in Group((s_{1,m_1}, ..., s_{n,m_n}), [\![q_Z]\!])(a_{1,m_1}, ..., a_{n,m_n})\}\!\} \end{gathered} \tag{22}$$

However, we know from Relation 20 how $Group((s_{1,m_1}, ..., s_{n,m_n}), [\![q_Z]\!])(a_{1,m_1}, ..., a_{n,m_n})$ looks like. We can use that to rewrite Relation 22 as:

$$\begin{gathered} multiset(a_{1,m_1}, ..., a_{n,m_n}) = \\ \{\!\{a'_{msr} \mid \exists a'_f :\ \{s_f \mapsto a'_f, s_{msr} \mapsto a'_{msr}, s_{1,m_1} \mapsto a_{1,m_1}, ..., s_{n,m_n} \mapsto a_{n,m_n}\} \in [\![q_Z]\!]\}\!\} \end{gathered} \tag{23}$$

We can also write Relation 23 as:

$$\begin{gathered} multiset(a_{1,m_1}, ..., a_{n,m_n}) = \\ \{\!\{a_{msr} \mid \exists a_f :\ \{s_f \mapsto a_f, s_{msr} \mapsto a_{msr}, s_{1,m_1} \mapsto a_{1,m_1}, ..., s_{n,m_n} \mapsto a_{n,m_n}\} \in [\![q_Z]\!]\}\!\} \end{gathered} \tag{24}$$

[3]Note that $Aggregation$ passes a multiset of lists to the function `f`. However, we transfer that to a multiset of values since the function $Flatten$ is called later in the SPARQL 1.1 specification for that purpose.

We know from Section 2.2.4 that the cardinality of each solution mapping in $[\![q_Z]\!]$ is **1** and that:

$$\begin{gathered} Z = \{(a_f, a_{msr}, a_{1,m_1}, ..., a_{n,m_n}) \in \mathcal{T}^{n+2} \mid \\ \exists \mu = \{s_f \mapsto a_f, s_{msr} \mapsto a_{msr}, s_1 \mapsto a_{1,m_1}, ..., s_n \mapsto a_{n,m_n}\} \in [\![q_Z]\!]\} \end{gathered} \tag{25}$$

Hence, the relation 24 can be written as:

$$multiset(a_{1,m_1}, ..., a_{n,m_n}) \;=\; \{\!\{a_{msr} \mid \exists a_f : (a_f, a_{msr}, a_{1,m_1}, ..., a_{n,m_n}) \in Z\}\!\} \tag{26}$$

From the definition of Z (Definition 9.16) we know that:

$$(a_f, a_{msr}, a_{1,m_1}, ..., a_{n,m_n}) \in Z \;\Leftrightarrow\; (a_f, a_{msr}) \in Y \wedge (a_f, a_{1,m_1}, ..., a_{n,m_n}) \in X \tag{27}$$

By applying the definitions of X (Definition 9.12) and Y (Definition 9.14) in Relation 27, it follows that:

$$\begin{gathered} (a_f, a_{msr}, a_{1,m_1}, ..., a_{n,m_n}) \in Z \;\Leftrightarrow \\ (a_f, a_{msr}) \in \lambda((f, msr)) \;\wedge a_f \in \mathscr{F} \;\wedge a_f \triangleright a_{1,m_1} \;\wedge\; ... \;\wedge\; a_f \triangleright a_{n,m_n} \end{gathered} \tag{28}$$

By applying Relation 28 in the definition of the function $multiset$ (Relation 26), it follows that:

$$\begin{gathered} mulitset(a_{1,m_1}, ..., a_{n,m_n}) = \{\!\{a_{msr} \mid \\ \exists a_f : (a_f, a_{msr}) \in \lambda((f, msr)) \;\wedge a_f \in \mathscr{F} \;\wedge a_f \triangleright a_{1,m_1} \;\wedge\; ... \;\wedge a_f \;\triangleright a_{n,m_n}\}\!\} \end{gathered} \tag{29}$$

According to the definitions of *grouping* and *aggregation* [1], [3], the final aggregation query (Definition 9.18) generates a mapping σ' defined as:

$$\begin{gathered} \sigma' = \{(a_{1,m_1}, ..., a_{n,m_n}) \mapsto multiset(a_{1,m_1}, ..., a_{n,m_n}) \mid \\ \exists a_f, \exists a_{msr} : \{s_f \mapsto a_f, s_{msr} \mapsto a_{msr}, s_{1,m_1} \mapsto a_{1,m_1}, s_{n,m_n} \mapsto a_{n,m_n}\} \in [\![q_Z]\!]\} \end{gathered} \tag{30}$$

Note that the previous relation can be written, using the same reasoning in Relation 25-Relation 29, as:

$$\begin{gathered} \sigma' = \{(a_{1,m_1}, ..., a_{n,m_n}) \mapsto multiset(a_{1,m_1}, ..., a_{n,m_n}) \mid \\ \exists a_f, \exists a_{msr} : (a_f, a_{msr}) \in \lambda(f, msr) \wedge a_f \in \mathscr{F} \wedge a_f \triangleright a_{1,m_1} \wedge ... \wedge a_f \triangleright a_{n,m_n}\} \end{gathered} \tag{31}$$

The mapping σ from Definition 9.9 is:

$$\begin{gathered} \sigma = \{(i_1, ..., i_n) \mapsto \mathscr{M}^{m_j}_{(i_1,...,i_n)} \mid (i_1, ..., i_n) \text{ is a coordinate of } (l_1, ..., l_n) \\ \text{and } \mathscr{M}^{m_j}_{(i_1,...,i_n)} \text{ is the respective non-empty multiset of the measure } m_j \text{ values}\} \end{gathered} \tag{32}$$

Which can be rewritten to:

$$\begin{gathered} \sigma = \{(i_1, ..., i_n) \mapsto \mathscr{M}^{m_j}_{(i_1,...,i_n)} \mid i_1 \in \lambda(l_1), ..., i_n \in \lambda(l_n) \\ \text{and } \mathscr{M}^{m_j}_{(i_1,...,i_n)} \text{ is the respective multiset of the measure } m_j \text{ values} \\ \text{and } \mathscr{M}^{m_j}_{(i_1,...,i_n)} \text{ is not empty } \} \end{gathered} \tag{33}$$

Which can be rewritten as:

$$\begin{gathered}\sigma = \{(i_1, ..., i_n) \mapsto \mathscr{M}^{m_j}_{(i_1,...,i_n)} \mid i_1 \in \lambda(l_1), ..., i_n \in \lambda(l_n) \\ \text{and } \mathscr{M}^{m_j}_{(i_1,...,i_n)} \text{ is the respective multiset of the measure } m_j \text{ values} \\ \text{and } \exists i_m, \exists i_f : (i_f, i_m) \in \lambda((f, m_j)) \wedge i_f \in \mathscr{F} \wedge i_f \triangleright i_1 \wedge ... \wedge i_f \triangleright i_n\}\end{gathered} \tag{34}$$

Note that $i_f \triangleright i_1 \wedge ... \wedge i_f \triangleright i_n$ implies that $i_1 \in \lambda(l_1), ..., i_n \in \lambda(l_n)$ (see Definition 9.5). Hence, Relation 34 can be rewritten as:

$$\begin{gathered}\sigma = \{(i_1, ..., i_n) \mapsto \mathscr{M}^{m_j}_{(i_1,...,i_n)} \mid \\ \text{and } \mathscr{M}^{m_j}_{(i_1,...,i_n)} \text{ is the respective multiset of the measure } m_j \text{ values} \\ \text{and } \exists i_m, \exists i_f : (i_f, i_m) \in \lambda((f, m_j)) \wedge i_f \in \mathscr{F} \wedge i_f \triangleright i_1 \wedge ... \wedge a_f \triangleright i_n\}\end{gathered} \tag{35}$$

By comparing the mappings: (i) σ from Relation 35 with (ii) σ' in Relation 31, and the multisets: (i′) $\mathscr{M}^{m_j}_{(i_1,...,i_n)}$ from Definition 9.9 with (ii′) $mulitset(a_{1,m_1}, ..., a_{n,m_n})$ in Relation 29, it follows that the aggregation query in Definition 9.18 correctly creates the mapping from coordinates to multisets of measures as specified per Definition 9.9. □

3 Properties of the Superimposed Multidimensional Schema

In this section, we prove some features over the MD schema of a superimposed MD schema.

Proposition 10. Assume a superimposed enriched MD model under the criteria established in Section 9.2.2 and given: (i) an arbitrary level $l_1 \in L$, (ii) an arbitrary level $l_2 \in L$, (iii) an arbitrary hierarchy $h \in H$, and (iv) an arbitrary hierarchy $h' \in H$, then the following holds. If $(h, l_1, l_2) \in HLL$ and $(h', l_1, l_2) \in HLL$, then $h = h'$, i.e., a roll-up association between two levels belongs to exactly one hierarchy.

Proof. In the following, we prove the proposition by showing that $h \neq h'$ leads to a contradiction. If $h \neq h'$, then $(h, l_2) \in HL$ and $(h', l_2) \in HL$ according to Definition 4.6 (*if there is a roll-up association between two levels in a hierarchy, then both of the levels should be in that hierarchy*). We know from Definition 4.6 that each hierarchy belongs to exactly one dimension and, hence, there are two cases: (I) h and h' are in the same dimension, and (II) h and h' are in different dimensions. In the following, we prove the proposition by showing that $h \neq h'$ leads to contradictions in both cases.

(I) h and h' are in the same dimension Let the dimension of h and h' be i. We know that $l_2 \neq top_i$ since we exclude the top levels of dimensions (Section 9.2.2.3). We also know that $l_2 \neq bot_i$ since there is no level that rolls up to bot_i according to Definition 4.8 (*each hierarchy $h \in H_i$ comprises a single, acyclic, loopless roll-up path that starts at bot_i and ends at top_i*). Hence, it follows that l_2 belongs to two different hierarchies h and h' (in the same dimension i) and l_2 is neither bot_i nor top_i, which contradicts Definition 4.8 (*except for bot_i and top_i, a level belongs to exactly one hierarchy in a dimension i*).

(II) h and h' are in different dimensions Assume that h_1 is in a dimension i and that h_2 in a dimension i' (different from i), i.e., (i) $(i, h) \in DH$, (ii) $(i', h') \in DH$, and (iii) $i \neq i'$. Hence, it follows that $(i, l_2) \in DL$ and $(i', l_2) \in DL$ (see the derivation of DL from DH and HL in Definition 4.7) which contradicts our assumption in Section 9.2.2.1 (*the sets of levels of any two different dimensions are disjoint*). □

Proposition 11. Assume a superimposed enriched MD model under the criteria established in Section 9.2.2 and the corresponding MD graph of that superimposed enriched MD model (Definition 9.1). There exists *at most* one MD path (Definition 9.2) between any two *nodes* of the MD graph, and there exists *at most* one MD path between any *node* of the MD graph and any *attribute* of the MD graph.

Proof. The proof is twofold: (A) node-to-node paths and (B) node-to-attribute paths. Details about MD graphs and MD paths can be found in Definitions 9.1 and 9.2, respectively. In the following, we may call an *MD path* a *path* for short.

(A) Node-to-Node Paths. Recall from Definition 9.1 that the nodes of the MD graph are the fact class and the levels of the MD model. A node-to-node path may be either a path from one level to another level or a path from the fact class to a level. Note that there exists no path from a level to the fact class since the fact class has only outgoing edges (Definition 9.1). For the same reason, there also exists no path from one level to another level where the path contains the fact class. Hence, in the following, we first show that there exists at most one path from one level to another, and then show that there exists at most one path from the fact class to a level.

Before we proceed, we make a few conventions. An MD path between two arbitrary levels l and l' is of the form $p = (l, (l, l_1), l_1, (l_1, l_2), l_2, ..., l_m, (l_m, l'), l')$ with $m \geq 0$ (Definition 9.2). We say that a level x is in the path p if $x \in p$. Similarly, we say that (x, y) is in the path p if $(x, y) \in p$. Note that all the pairs in p are roll-up associations since a node-to-node path comprises edges, which are only from $LL^- \cup FL$, and we have already shown that the fact class cannot be in a path from one level to another.

In order to show that there exists at most one path from a level to another level, we follow three steps:

- First, we show that: (I) if a path exists between l and l', then all the levels in the path are in the same dimension.
- Second, we show that: (II) if a path exists between l and l', then all the levels in this path are in the same hierarchy.
- Finally, we show that: (III) the existence of more than one path between l and l' leads to a contradiction, which proves that there exists at most one path between any two arbitrary levels.

We detail these steps in the following.

(I) If a path exists between l and l', then all the levels in the path are in the same dimension. Let the existing path that connects l and l' be p, and let us take an arbitrary pair of levels $(l_x, l_y) \in LL$ such that (l_x, l_y) is in the path p. Let l_x belong to a dimension i_x, i.e., $(i_x, l_x) \in DL$,

and let l_y belong to a dimension i_y, i.e., $(i_y, l_y) \in DL$. We know from Definition 4.6 that *if there is a roll-up association between two levels in a hierarchy, then both of the levels must be in that hierarchy*. Hence, both l_x and l_y are in the same hierarchy; let that hierarchy be h, i.e., $(h, l_x) \in HL$ and $(h, l_y) \in HL$. The hierarchy h is *unique* according to Proposition 10. The fact that $(h, l_x) \in HL$ and $(h, l_y) \in HL$ implies that there exists some dimension $i \in D$ such that $(i, h) \in DH$ according to Definition 4.6 (*each hierarchy belongs to exactly one dimension*), which implies that $(i, l_x) \in DL$ and that $(i, l_y) \in DL$ (see Definition 4.7). Since $(i, l_x) \in DL$ and $(i_x, l_x) \in DL$ and we assumed that *the sets of levels of any two different dimensions are disjoint* (Section 9.2.2.1), it follows that $i = i_x$. Similarly, since $(i, l_y) \in DL$ and $(i_y, l_y) \in DL$ and we assumed that *the sets of levels of any two different dimensions are disjoint* (Section 9.2.2.1), it follows that $i = i_y$, which also implies that $i_x = i_y$. Hence, it follows that the two levels in any pair of levels (l_x, l_y) in the path p are in the same dimension. Since a level belongs only to one dimension according to Section 9.2.2.1 (the assumption that *the sets of levels of any two different dimensions are disjoint*), it follows that all the levels in the path belong to the same dimension.

(II) If a path exists between l and l', then all the levels in this path are from the same hierarchy. Let the existing path that connects l and l' be p, and let us take an arbitrary pair of levels $(l_x, l_y) \in LL$ such that (l_x, l_y) is in the path p. We know from Definition 4.8 that, given a dimension i, *a level that is different from bot_i and different from top_i belongs to exactly one hierarchy in the dimension i*. Furthermore, and since we exclude top levels of dimensions (Section 9.2.2.3), only the level bot_i may belong to multiple hierarchies in the dimension i. We know that *a hierarchy h in a dimension i constitutes an single, acyclic, and loopless path from bot_i to top_i* (Definition 4.8). Note that l_x and l_y are in the same hierarchy according to Definition 4.6 (*if there is a roll-up association between two levels in a hierarchy, then both of the levels must be in that hierarchy*). This "same hierarchy" of l_x and l_y is the hierarchy of l_y since l_y is in exactly one hierarchy whereas l_x can be in multiple hierarchies because it may be the case that $l_x = bot_i$. Thus, we have shown that for an arbitrary pair of levels (l_x, l_y) in the path p, l_x and l_y are in the hierarchy of l_y. Given that conclusion and since *any level in the path p is in exactly one hierarchy in the dimension it belongs to (except for the start level l, which may belong to multiple hierarchies in the dimension it belongs to)* (see Definition 4.8) and also since *any hierarchy belongs to exactly one dimension* (Definition 4.6), it can be easily concluded that all the levels in the path belong to a single hierarchy; that single hierarchy is the hierarchy of any level in the path except for the start level l.

(III) The existence of more than one path between l and l' leads to a contradiction. First, recall that *any hierarchy belongs to exactly one dimension* (Definition 4.6). Assume that there exist two paths p_1 and p_2 between l and l'. According to Step (I), all the levels in both paths p_1 and p_2 are in the same dimension; let that dimension be i. The reason behind that is that levels in p_1 and levels in p_2 are in two identical dimensions according to the assumption in Section 9.2.2.1 (*the sets of levels of any two different dimensions are disjoint*), since both paths contain l. Furthermore, and according to Step (II), all the levels in p_1 belong to an identical

hierarchy; let that hierarchy be h_1, and all the levels in p_2 belong to an identical hierarchy; let that hierarchy be h_2. Note that h_1 and h_2 must be in the dimension i of the levels in the paths p_1 and p_2 (otherwise, levels in h_1 and h_2 will be in two different dimensions, which contradicts the assumption that in Section 9.2.2.1 that *the sets of levels of any two different dimensions are disjoint*). There exist two possible cases: either $h_1 = h_2$ or $h_1 \neq h_2$. The level l' is shared between both paths p_1 and p_2, which implies that l' is in both hierarchies h_1 and h_2. If $h_1 \neq h_2$, then l' belongs to two hierarchies in the same dimension, which results in a contradiction. In particular, only the level bot_i of a dimension may belong to multiple hierarchies since *a level different from bot_i and top_i belongs to exactly one hierarchy in the dimension i* (Definition 4.8) and since we exclude the top levels of dimensions (Section 9.2.2.3). As a consequence, $h_1 = h_2$ holds, which leaves us with two paths p_1 and p_2 between l and l' where all the levels of both paths are in one hierarchy $h_1 = h_2$. That results in a contradiction since *a hierarchy h in a dimension i constitutes an single, acyclic, and loopless path from bot_i to top_i* (Definition 4.8). Hence, the existence of more than one path between l and l' leads to a contradiction, which proves that there exists at most one path between any two arbitrary levels.

The second type of node-to-node paths is paths from the fact class to a level. We have proved that there exists at most one path between one level and another, and we have shown that the levels of this path are in the same dimension and the same hierarchy. We also know that the fact class is directly connected to the bottom level of each dimension. Hence, it intuitively follows that there exists at most one path between the fact class and any level of any dimension.

(B) Node-to-Attribute Paths. The second type of paths is paths that connect the fact class to a base measure or paths that connect a fact or a level to an attribute. In the first part of this proof, we have proved that there exists at most one path between any two nodes of the MD graph. Consider an arbitrary node e of the MD graph and an arbitrary attribute of the MD graph (e', a). If a path exists between e and (e', a), then this path is of the form $p = (e, (e, e_1), e_1, (e_1, e_2), e_2, ..., e_m, (e_m, e'), e', (e', a))$, which starts in the node e and has the node e' as the second last element (Definition 9.2)[4]. Since there exists at most one path between e and e' according to what we have proved in the first part of this proof (the path is of the form $(e, (e, e_1), e_1, (e_1, e_2), e_2, ..., e_m, (e_m, e'), e')$), it follows that there exists at most one path between e and (e', a) (the path is of the form $(e, (e, e_1), e_1, (e_1, e_2), e_2, ..., e_m, (e_m, e'), e', (e', a))$). The special case of Definition 9.2 is when e is a fact and a is a measure, or when e is a level and a is an attribute of that level. In that special case, it is obvious that there exists at most one path between e and (e, a), where we have that $e = e'$ and the path is of the form $(e, (e, a))$. □

[4]Recall that, in the MD graph, an attribute $(e, a) \in LA \cup FM$ is determined by its source node $e \in F \cup L$ and its attribute $a \in A$ (i.e., not only by $a \in A$).

Bibliography

[1] W3C, *SPARQL 1.1 Query Language: W3C Recommendation 21 March 2013*, S. Harris and A. Seaborne, Eds., 2013. [Online]. Available: https://www.w3.org/TR/2013/REC-sparql11-query-20130321/ (visited on 17/12/2023).

[2] M. Schmidt, M. Meier and G. Lausen, 'Foundations of SPARQL query optimization,' in *Database Theory ICDT 10, 13th International Conference, Lausanne, Switzerland, March 23-25, 2010, Proceedings*, ser. International Conference Proceeding Series, L. Segoufin, Ed., Lausanne, Switzerland: ACM, 2010, pp. 4–33, ISBN: 978-1-60558-947-3. DOI: 10.1145/1804669.1804675.

[3] M. Kaminski, E. V. Kostylev and B. C. Grau, 'Query nesting, assignment, and aggregation in SPARQL 1.1,' *ACM Transactions on Database Systems (TODS)*, vol. 42, no. 3, 17:1–17:46, 2017. DOI: 10.1145/3083898.

[4] J. Salas and A. Hogan, 'Semantics and Canonicalisation of SPARQL 1.1,' *Semantic Web*, vol. 13, no. 5, pp. 829–893, 2022. DOI: 10.3233/SW-212871.

[5] A. Polleres and J. P. Wallner, 'On the relation between SPARQL1.1 and Answer Set Programming,' *Journal of Applied Non-Classical Logics*, vol. 23, no. 1-2, pp. 159–212, 2013. DOI: 10.1080/11663081.2013.798992.

[6] D. Colazzo, F. Goasdoué, I. Manolescu and A. Roatiş, 'RDF Analytics: Lenses over Semantic Graphs,' in *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014*, C.-W. Chung, A. Z. Broder, K. Shim and T. Suel, Eds., ACM, 2014, pp. 467–478, ISBN: 978-1-4503-2744-2. DOI: 10.1145/2566486.2567982.

[7] J. Pérez, M. Arenas and C. Gutierrez, 'Semantics and complexity of sparql,' *CoRR*, vol. abs/cs/0605124, 2006. [Online]. Available: http://arxiv.org/abs/cs/0605124.