Extending the Concept of Voting Structures to Support Path-Based Replication Strategies: the Proofs

Christian Linder and Oliver Theel
Department of Computer Science
Carl von Ossietzky University
Oldenburg, Germany
Email: christian.linder@uni-oldenburg.de,
oliver.theel@uni-oldenburg.de

September 2021

Technical Report
TR-SVS-2021-01

Data replication is an important concept in distributed computing to provide fault-tolerant, highly available, and low cost data access operations, since a unique master copy can fail. Although the field of data replication is very well researched, the question of application-optimized replication strategies for specific scenarios still stands. There are many scenarios which cannot be satisfactorily managed by known replication strategies. Recent research has proposed the use of voting structures for generating new replication strategies from known ones, such that a wide range of scenarios can be addressed. Since voting structures could not represent path-based replication strategies in an intuitive way, the concept of voting structures has recently been extended to support path-based replication strategies in an easy way. Furthermore, a general design pattern has been presented that transforms each path-based replication strategy, given by a grid structure, to an efficient equivalent voting structure of the extended concept. Since path-based replication strategies have been proven to be very flexible and efficient in providing low operation costs and high availabilities, some very good characteristics can now be covered by the above-sketched approach. In this technical report, we present the *formal proofs* of the correctness for both the concept of *extended* voting structures as well as the general design pattern.

1 Introduction

Voting structures form a concept in the field of data replication, which allows to design different quorum-based replication strategies in an easy, efficient, and intuitive way. In general, data replication is a concept to increase the availability of data access operations degraded by failures, thus making them more fault-tolerant. This is done by distributing multiple data copies (i.e., replicas) on different computers. While improving availability, the use of replicas can result in low but also quite high operation costs depending on the particular circumstances. Replication strategies manage and synchronize data access to the replicas, thereby achieving different trade-offs between availability, costs, and data consistency. We only consider replication strategies which ensure a certain notion of data consistency. Such replication strategies provide a single copy image of the replicated data, i.e., data access operations behave the same way on replicated data as they do on non-replicated data. A common formal correctness criterion for such a behavior is called one-copy-serializability (1SR) [1]. Many such replication strategies fulfilling 1SR are known from literature. They are broadly divided into two classes: unstructured ones such as Majority Consensus (MCS) [2], Read one / Write all [3], Quorum Systems [4], and structured ones such as the Grid Protocol [5], or the Triangular Lattice Protocol (TLP) [6]. Unstructured replication strategies are mostly based on votes and quorums, whereas structured replication strategies are bases on logical patterns. These strategies all achieve a certain compromise between operation costs and availability, thereby addressing different application scenarios.

Since there is no *one* best quorum-based replication strategy for *all* application scenarios [7], it is desirable to be able to find an optimal replication strategy in a particular scenario specified by requirements to availability and costs. Although there are already some hybrid approaches that combine known replication strategies to obtain new trade-offs between availability and costs [8,9], they can only cover a very small part of possible scenarios. Therefore, the question of finding application-optimized replication strategies must more generally be answered. Due to new possibilities in the field of computational intelligence, this question is very current in combination with established concepts in the problem domain. Recent research has proven voting structures to be very suitable to automate the process of discovering such scenario-based application-optimized replication strategies through genetic programming [10]. This approach is very promising, but, unfortunately, —so-called— *path-based* replication strategies cannot appropriately be presented as a voting structure. However, path-based replication strategies have proven to be very flexible and efficient in providing low operation costs and high availability. Therefore, we want to take advantage of the efficiency of path-based replication strategies to improve the aforementioned new approach for finding application-optimized replication strategies at a large scale.

In [11], we have presented an extension to the concept of voting structures as well as a design pattern to model any path-based replication strategy as a so-called *extended* voting structure. In [11], we could only state proof *ideas* for the correctness of the presented concepts due to space limitations. In this technical report, therefore, we provide the *formal proofs* of correctness for both the concept of *extended* voting structures and

the general design pattern.

After summarizing related work in Section 2, we recap the definitions of the extended concept of voting structures in Section 3 and the design pattern of path-based replication strategies in Section 4. In Section 5, we introduce derivation trees formally, which are necessary for the correctness proofs in Section 6. Finally, we draw a conclusion of this report in Section 7.

2 Related Work

We now refer to related work on voting structures and path-based replication strategies.

First, we review the concept of quorums to raise an understanding of voting structures as well as path-based replication strategies. Quorum systems are a general approach to define replication strategies. The idea is to list all combinations of replicas that allow 1SR-conform operation execution under the assumption of crash-failure semantics for the replicas. For this purpose, a set of read quorums and a set of write quorums is defined. Each read quorum (write quorum) consists of one or more replicas that must be used to consistently perform a read (write) operation on the replicated data object. The two sets are constructed in such a way that each read quorum overlaps with each write quorum, and each write quorum overlaps with every other write quorum in at least one replica. [4]

Example 1. *Consider a distributed system with the replicas* R_1, R_2, *and* R_3. *Then, the set of read quorums* $Q_r = \{\{R_1\}, \{R_2, R_3\}\}$ *and the set of write quorums* $Q_w = \{\{R_1, R_2\}, \{R_1, R_3\}\}$ *form a sample quorum system, since they fulfill the mentioned overlapping properties.*

Quorum systems can serve as a common base for originally different representations of replication strategies. Thus, they allow voting structures and path-based replication strategies to be compared easily and intuitively.

2.1 Voting Structures

General Structured Voting (GSV) [12] is a flexible modeling framework for replication strategies that uses so-called *voting structures*. Furthermore, GSV provides a uniform implementation that brings a replication strategy to life, based on a given voting structure. In that way in GSV, the development of replication strategies is reduced to the modeling of voting structures. The great strength of voting structures is that they have proven to be flexible enough to represent *any* quorum-based replication strategy and many of these can be modeled in a very intuitive and elegant way. A voting structure is a graph of physical and virtual nodes using votes and scalar operation quorums to derive read and write quorums. The graph must be coherent, acyclic, and must have a unique root node. Due to their graphical nature, voting structures are well understandable for humans and they can reduce, in general, the memory complexity compared to quorum systems. For many replication strategies, it is not difficult to find corresponding voting

structures [12]. Moreover, GSV can be used as a graphical design tool for designing and combining different replication strategies [13].

Figure 1 shows an example of a voting structure for a distributed system with three computers with replica, labeled R_1, R_2, and R_3. The computers with replica of the

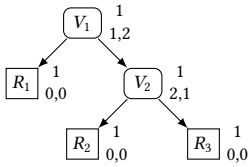

Figure 1: Example of a Voting Structure [11]

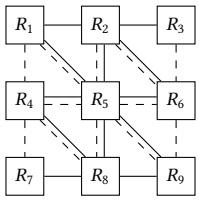

Figure 2: Grid Structure of TLP 3×3 [14]

distributed system are called *physical nodes* in GSV. All other nodes used in a voting structure are called *virtual nodes* (V_1 and V_2 in the example). Physical nodes are represented by a square and labeled with the identifier of the corresponding computer with replica. Virtual nodes are represented by boxes with rounded corners and a unique label (e.g., V_1, but any label is allowed). On the right side of these boxes, representing virtual and physical nodes, *votes* and *scalar*[1] *operation quorums* are given. The upper index represents the vote, the lower indices represent the scalar operation quorums. Unless specified otherwise, the scalar read quorum is on the left side and the scalar write quorum is on the right side. The nodes are connected by directed edges. If there is an edge $V_1 \rightarrow V_2$, then V_2 is called *direct successor* of V_1. The *unique root node* in the figure is V_1, since V_1 is no direct successor of any other node.

To interpret such a voting structure, GSV derives read and write quorums in the following way. To derive a read quorum (write quorum), the root node must collect enough votes of its direct successors to reach its scalar read quorum (scalar write quorum). For inner nodes, to pass their votes up, the same procedure is (recursively) used. Physical nodes *additionally* require that the corresponding computer with replica is available and that the replica is not already locked for a conflicting operation. Usually, the *leaves* (nodes without successor) are physical nodes with scalar quorums of *zero*, and then, they vote *iff* the computer with replica is available and the replica is not already locked for a conflicting operation. A single read quorum (write quorum) can be derived, if the root node votes, and then it consists of all physical nodes that have given their votes within the above-sketched (recursive) procedure.

The voting structure in Figure 1 derives the same quorums given by the quorum system in Example 1. The problem of locked replicas is not essential for the derivation of quorums, as it only arises when an operation is actually intended to be executed. To derive a read quorum, e.g., V_1 needs the scalar read quorum of *one*. Thus, the vote *one* of R_1 *or* the vote *one* of V_2 is enough. Therefore, $\{R_1\}$ is a read quorum as well as the derived read quorums of V_2. V_2, though, needs the scalar read quorum of *two*. Thus,

[1]Note the difference between *scalar* read quorum and read quorum resp. *scalar* write quorum and write quorum. This distinction is made consistently in the remainder of this paper.

the vote *one* of R_2 *and* the vote *one* of R_3 are necessary. Therefore, $\{R_2, R_3\}$ is also a read quorum.

2.2 Path-Based Replication Strategies

The Triangular Lattice Protocol (TLP) [6] is an example of a *path-based replication strategy (PB-RS)*. TLP uses nodes, arranged logically in a grid structure, which are connected by edges, such that the resulting graph forms a triangular lattice. The concept and idea was presented in [6] and a generalization of TLP and further PB-RS was presented in [14].

The basic idea of general PB-RS, which describes one possible class of *structured* replication strategies, is very similar to the idea of the original TLP. The nodes of a distributed system are arranged logically in a grid structure, which are connected by undirected edges. Figure 2 shows the grid structure of TLP 3×3 (i.e., TLP with *9* replicas arranged logically in a grid with *3* columns and *3* rows). There are two kind of edges, namely *horizontal edges* (represented by solid lines) and *vertical edges* (represented by dashed lines). A *horizontal (vertical) path* starts at a node of the leftmost column (top row) and uses only horizontal (vertical) edges to reach the rightmost column (bottom row). We represent paths by the notation $\langle v_1, \ldots, v_m \rangle$, if there is an edge between each v_i and v_{i+1} for $1 \leq i \leq m-1$. By using these paths, quorums are derived. A read quorum must contain at least the nodes of a horizontal *or* a vertical path. A write quorum must contain at least the nodes of a horizontal *and* a vertical path. [14]

$\langle R_4, R_5, R_9 \rangle$ is an example for a horizontal path in TLP 3×3 (cf. Figure 2). The path $\langle R_3, R_6, R_5, R_4, R_7 \rangle$ is an example of a vertical path. $\langle R_1, R_5, R_9 \rangle$ is an example of a vertical *and* of a horizontal path. Thus, we have some examples of read quorums: $\{R_4, R_5, R_9\}$, $\{R_3, R_4, R_5, R_6, R_7\}$, $\{R_1, R_5, R_9\}$, and we have examples of write quorums: $\{R_3, R_4, R_5, R_6, R_7, R_9\}$, $\{R_1, R_5, R_9\}$.

The concept of PB-RS forms a design framework for a whole class of replication strategies with very good characteristics, in particular wrt. operation costs.

2.3 Comparison of the Concepts

Both, the concept of voting structures and the concept of PB-RS are modeling frameworks in the context of data replication. PB-RS are efficient in providing low operation costs with only slightly reduced but still high availability, in particular, compared to unstructured replication strategies (e.g. MCS). However, they are far from being able to represent all quorum-based replication strategies. This, on the other hand, is what voting structures can realize [12]. In addition, they can also be used very flexibly to compose known replication strategies into hybrid ones. A flexible hybrid approach using voting structures has been presented in [9]. It has been supplemented by genetic programming in [10]. In this way, *application-optimized* replication strategies can be found automatically by means of scenario parameters setting requirements wrt. the desired availability and costs. A scenario can be given, e.g., by the following parameters:

- Number of replicas: 16

- Availability of single replicas: 0.9
- Min. read availability of the replication strategy: 0.996
- Min. write availability of the replication strategy: 0.995
- Max. write cost of the replication strategy: 3.10
- Max. read cost of the replication strategy: 3.12

In order to find a replication strategy fulfilling these requirements, known replication strategies, given by voting structures, are refined by applying crossover and mutation operations. These voting structures are evaluated against the scenario parameters and continuously further evolved.

It would be very useful to be able to use voting structures for PB-RS in the above-stated approach, in order to render their good characteristics easily available to the combination of various voting structures. Certainly, it is possible to represent PB-RS as a voting structure, e.g., by modeling the corresponding quorum system. However, this approach is not appropriate, especially in terms of scalability and comprehensibility, due to its set-like nature with exponentially many elements [15].

3 Extended Voting Structures

In this section, we recap the extension [11] of the concept of voting structures [12]. First, we present an adapted and simplified[2] model, that allows cycles in the graph of a voting structure. Secondly, we present the adapted algorithm for deriving quorum sets, such that cyclic graphs do not lead to a non-terminating computation.

3.1 Model

Now, we present our syntactical model of an extended voting structure and some additional definitions to the concept presented in [12], thereby explain the differences to the original model when necessary.

Definition 1 (Extended Voting Structure [11])**.** *Let the set* $R = \{R_1, \ldots, R_n\}$ *contain the computers with replica of a distributed system. Then,* $\mathrm{VS}^+ := (V, \lhd, w, \underline{v}, \underline{s}, \underline{q_{op}})$ *defines an* extended voting structure (EVS) *with nodes* $V \supseteq R$ *and directed edges* $\lhd \subseteq V \times V$*, whose graph has to be finite, coherent, and loop-free. There is a unique* root *node* $w \in V$ *for which* $\forall w' \in V : ((\nexists\, v \in V \;:\; v \lhd w') \Leftrightarrow w' = w)$ *holds. The nodes* $R_i \in R$ *are called* physical nodes*. The nodes* $V_i \in V \setminus R$ *are called* virtual nodes*.* VS^+ *includes a tuple* $\underline{v}$*, whose elements represent all nodes of* V *in an arbitrary, but fixed total order. It also includes a tuple* $\underline{s}$ *and tuples* $\underline{q_{op}}$*, for each operation* op*, with the same dimension as* $\underline{v}$*. The element* $\underline{s}_i \in \mathbb{N}$ *is called the* vote *of the node* $\underline{v}_i$ *and* $\underline{q_{op}}_i \in \mathbb{N}_0$ *is called the* scalar operation quorum *of the node* $\underline{v}_i$ *for operation* op*.*

[2]Here, we omit the definition of edge weights of a voting structure, which are used to prioritize the derived quorums. Prioritization is an "add-on" for voting structures. It is not essential for the main result of this paper. We have presented an extension of edge weights in [16].

The only change to the original concept is, that we no longer require the graph of an EVS to be acyclic, but only loop-free instead, i.e., that there is no edge from a node to itself. Additionally, the unique root node w, the sequence $\underline{v}$, the votes $\underline{s}$, and the scalar operation quorums $\underline{q_{op}}$ are added to the tuple-like definition of VS^+ for an easier interface definition of the algorithm (see Section 3.2). Virtual nodes are identified by arbitrary but unique names.

Figure 3 shows an example of an EVS, which is not a valid voting structure of the original concept. In the extended concept, if there are edges between two nodes in

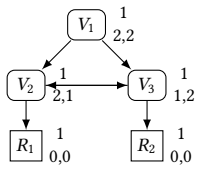

Figure 3: Example of an Extended Voting Structure [11]

both directions, then this is represented by a bidirectional arrow. Obviously, the edge between V_2 and V_3 is the reason, why this EVS is not a valid voting structure of the original concept, since this edge leads to a cycle.

Now, we give some additional definitions to facilitate the handling of EVS.

Definition 2 (k-Minimality [12])**.** *A set of nodes is* k-minimal*, if the sum of the votes, owned by the nodes, is greater or equal to* k*, but without the vote of any of these nodes, the sum is less than* k*. Accordingly, a node* v *is called* k-minimal*, if the set* $\{v\}$ *is* k*-minimal.*

Definition 2 has been adopted for the extended concept as it is. It provides a criterion to describe that a node achieves its scalar operation quorum by a subset of its direct successors, which pass their votes up. For this case, the number k is set to the scalar operation quorum of the node. Only minimal necessary voting combinations are allowed per node.

Definition 3 (Leaf Node [11])**.** *Let* $\mathrm{VS}^+ = (V, \lhd, w, \underline{v}, \underline{s}, \underline{q_{op}})$ *be an EVS. The node* $v \in V$ *is a* leaf *of* VS^+*, iff* $\nexists\, v' \in V \;:\; v \lhd v'$*.*

Definition 4 (Node Index [11])**.** *Let* $\mathrm{VS}^+ = (V, \lhd, w, \underline{v}, \underline{s}, \underline{q_{op}})$ *be an EVS and* $v \in V$ *be a node of* VS^+*. The function* $idx : V \to \mathbb{N}$ *provides the* index *of* v *in* VS^+ *where* $idx(v) = i :\Leftrightarrow v = \underline{v}_i$*.*

Definitions 3 and 4 are used to describe the following algorithm in a more concise way.

3.2 Algorithm

With the model of extended voting structures and the above-stated additional definitions, an algorithm $quorums^+_{op}$ can be specified which calculates the set of quorums for each operation.

The base of the algorithm in Figure 4 has been taken from [12] and it has been renamed and adapted to the model, defined in the previous section. It has been extended by the parameter $Pre \subseteq V$, which is used in Line 9 and 10 to prevent the occurrence of infinite sequences of recursion steps in the derivation of quorums. Additionally, the compose operation in Line 11 has been adapted.

```
1  quorums^+_op(in: VS^+, a, Pre): QS_op {
2    Q_a := ∅
3    k := idx(a)
4    if (a is a leaf of VS^+) {
5      if (a ∈ R)
6        Q_a := Q_a ∪ {{a}}
7    }
8    else {
9      for all (b_i, i = 1,...,l with a ◁ b_i ∧ b_i ∉ Pre) do
10       Q_{b_i} := quorums^+_op(in: VS^+, b_i, Pre ∪ {a})
11     L := { ⋃_{i∈I} q_i | ⋀_{i∈I} q_i ∈ Q_{b_i} ∧ ⋃_{i∈I} {b_i} is q_{op_k}-minimal ∧ I ⊆ {1,...,l} }
12     for all (m ∈ L) do
13       if (a ∈ R)
14         Q_a := Q_a ∪ {m ∪ {a}}
15       else
16         Q_a := Q_a ∪ {m}
17   }
18   return Q_a
19 }
```

Figure 4: New Algorithm for the Calculation of Quorums [11] based on [12]

The algorithm derives *all* quorums of the operation *op* for a given EVS. A detailed explanation of the algorithm's functioning can be found in [11].

4 Design Pattern for Path-Based Replication Strategies

By using the extension of voting structures, as presented in the previous section, we can now model voting structures for PB-RS. Before specifying the design pattern, we adopt the definition of grid structures used in [14], since it provides us with a representation of arbitrary PB-RS. A grid structure is formally defined by a graph of nodes and two kind of edges. Quorums are derived by finding paths in such a graph.

Definition 5 (Grid Structure [14])**.** *Let R be the set of nodes of a distributed system. The tuple $GS := (R, \{\Box, \diamond\}, S_\Box, S_\diamond, E_\Box, E_\diamond)$ defines a* grid structure*. $\Box \subseteq R \times R$ is called the set of* horizontal edges *and $\diamond \subseteq R \times R$ is called the set of* vertical edges*. $S_\Box \subseteq R$ is called* horizontal start set *and $E_\Box \subseteq R$ is called* horizontal end set*. Analogously, $S_\diamond, E_\diamond$ are called* vertical start set *and* vertical end set*.*

Definition 6 (Quorums in a Grid Structure [14])**.** *Let $GS := (R, \{\Box, \diamond\}, S_\Box, S_\diamond, E_\Box, E_\diamond)$ be a grid structure. A path $\langle p_1, \ldots, p_m \rangle$ with $p_i \in R$ for $1 \leq i \leq m$, $p_1 \in S_\Box$, $p_m \in E_\Box$,*

and $(p_i, p_{i+1}) \in \square$ *for* $1 \leq i \leq m-1$ *is called* horizontal path. *A* vertical path *is defined analogously with* $\diamond$ *instead of* $\square$. *A path* must not *contain the same node more than once. A* read *quorum must contain the nodes of a vertical* or *a horizontal path. A* write *quorum must contain the nodes of a vertical* and *a horizontal path.*

Based on the definitions of grid structures, we now define a design pattern to transform any PB-RS into an equivalent voting structure. We denote the universe of all possible grid structures with U_{GS} and the universe of all possible EVS with U_{VS^+}.

Definition 7 (Design Pattern for PB-RS [11])**.** *Let* $GS = (R, \{\square, \diamond\}, S_\square, S_\diamond, E_\square, E_\diamond)$ *be a grid structure and, w.l.o.g.,* $R = \{R_1, \ldots, R_n\}$. *Then,* $VSP : U_{GS} \to U_{VS^+}$ *with* $VSP(GS) := \text{VS}^+ = (V, \lhd, w, \underline{v}, \underline{s}, \underline{q_{op}})$ *is the* Design Pattern for PB-RS *where* VS^+ *uses the labels* G, V^0, H^0, V_i, H_i *with* $1 \leq i \leq n$ *for the virtual nodes and is defined as follows:*

$$V := \{G, V^0, H^0\} \cup \{V_i, H_i \mid 1 \leq i \leq n\} \cup R \tag{1}$$

$$\lhd := \{(G, V^0), (G, H^0)\} \tag{2}$$

$$\cup \{(V^0, V_i) \mid R_i \in S_\diamond\} \cup \{(H^0, H_i) \mid R_i \in S_\square\} \tag{3}$$

$$\cup \begin{array}{l} \{(V_i, V_j) \mid (R_i, R_j) \in \diamond\} \cup \\ \{(H_i, H_j) \mid (R_i, R_j) \in \square\} \end{array} \tag{4}$$

$$\cup \{(V_i, R_i), (H_i, R_i) \mid 1 \leq i \leq n\} \tag{5}$$

$$w := G \tag{6}$$

$$\underline{v} := \begin{array}{r} (G, V^0, H^0, V_1, \ldots, V_n, H_1, \ldots, H_n, \\ R_1, \ldots, R_n) \end{array} \tag{7}$$

$$\underline{s}_i := \begin{cases} n & \textit{if } \underline{v}_i \in R \\ 1 & \textit{otherwise} \end{cases} \tag{8}$$

$$\underline{q_{op}}_i := \begin{cases} 0 & \textit{if } \underline{v}_i \in R \\ n+1 & \textit{if } \underline{v}_i \in \{V_k \mid R_k \notin E_\diamond\} \cup \\ & \qquad \{H_k \mid R_k \notin E_\square\} \\ n & \textit{if } \underline{v}_i \in \{V_k \mid R_k \in E_\diamond\} \cup \\ & \qquad \{H_k \mid R_k \in E_\square\} \\ 2 & \textit{if } \underline{v}_i = G \ \wedge \ op = \textit{write} \\ 1 & \textit{otherwise} \end{cases} \tag{9}$$

For the remainder of this paper, we refer to Definition 7 as the *design pattern.* The design pattern and the grid structure of TLP 3×3 (cf. Figure 2) provide the EVS for TLP 3×3 shown in Figure 5. To render the modeling more descriptive, scalar quorums of *ten* are indicated by an "x" due to space limitations. Additionally the edges $V_i \to R_i$ and $H_i \to R_i$ for all $1 \leq i \leq 9$ are only schematically indicated. Otherwise, readability of the figure would get lost. However, these adaptations only affect the representation, but not the functioning of the voting structure. We have taken advantage of the possibility

to label virtual nodes with arbitrary names in order to raise understanding of the idea of the particular voting structure.

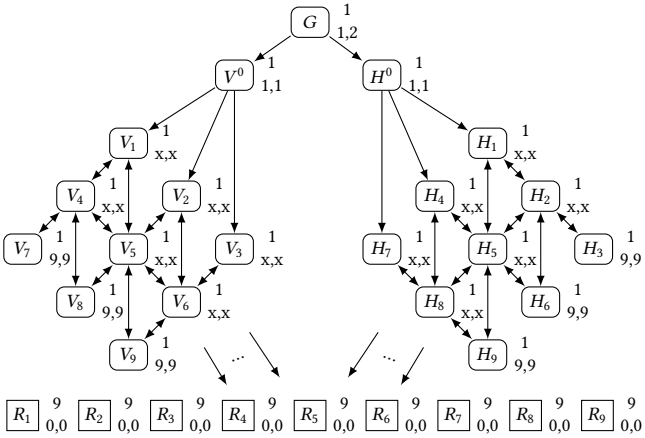

Figure 5: Extended Voting Structure for TLP 3×3 [11]

With the given design pattern it is possible to model an EVS for any PB-RS given by a grid structure. A detailed explanation of the functioning of the design pattern can be found in [11].

5 Derivation Trees

In Section 2.1, we have explained the intuitive procedure for deriving a *single* quorum of a voting structure. By following the algorithm of Figure 4, we have a formal definition of how *all* quorums of an EVS are derived per operation. In order to formalize the explained intuitive procedure, we now introduce the concept of a *derivation tree* as a graphical representation of a single algorithm execution. In addition, we supplement the concept with the definition of a *local* derivation tree *(LDT)* which we need to prove the relation between a derivation tree and the calculation of the algorithm.

Figure 6 shows an example of a derivation tree D of the EVS shown in Figure 3 for the quorum $q = \{R_1, R_2\}$ in operation *read*. Figure 7 shows an example LDT L of the same EVS for $\tilde{q} = \{R_1\}$ of the operation *read*. Note, that although votes and scalar operation quorums are not formally part of a (local) derivation tree, we include them in the graphical representation for ease of understanding. In a derivation tree, each recursive invocation of the algorithm $quorums^{+}_{op}$ is represented by a node. With a function *src*, each node of the derivation tree can be assigned to its *source* in the corresponding EVS, which is the parameter a of its invocation, from which it has been originated. This means, a node of the derivation tree serves as a *proxy* of a node of its corresponding EVS. In the sample derivation tree D, the function *src* is implicitly defined by the names of the nodes, which are equal to the names of their sources with one or more additional primes (e.g., $src(V_3') = V_3$ and $src(R_2'') = R_2$). We call V_3 the *source* of

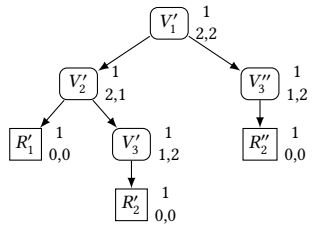

Figure 6: Examples of a Derivation Tree

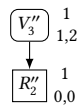

Figure 7: Example of a LDT

V_3' and, vice versa, we call V_3' a *proxy* of V_3. By this particular derivation tree, one can deduce that the nodes V_3 and R_2 have each been visited twice by the algorithm execution under consideration and that these visits can easily be distinguished by the one or two prime symbols, respectively. Additionally, the derivation tree contains exactly those nodes whose sources have been visited by the algorithm execution under consideration to calculate a *single* quorum q of the total result of algorithm $quorums^{+}_{op}$.

Compared to a derivation tree, an LDT can have a root node whose source is not the root of the corresponding EVS.

Now, we define the concept of a derivation tree formally. Afterwards, we explain the meaning of the definition step by step and justify that the graph of D in Figure 6 exhibits all the required properties of a derivation tree. Finally, we define the concept of an LDT formally.

Definition 8 (Derivation Tree)**.** *Let $VS^{+} = (V, \lhd, w, \underline{v}, \underline{s}, \underline{q_{op}})$ be an extended voting structure and $q \subseteq R$ a quorum of* VS^{+}*. A tree $D = (V_D, \lhd_D)$ with root w_D is called a* derivation tree *of* VS^{+} *for the quorum* q *in operation* op*, if there exists a function* $\mathrm{src} : V_D \to V$ *for which the following holds.*

1. $\forall v_1, v_2 \in V_D : \; v_1 \lhd_D v_2 \Rightarrow \mathrm{src}(v_1) \lhd \mathrm{src}(v_2)$
2. $\forall v_1, v_2 \in V_D : \; \exists \; \langle v_1, \ldots, v_2 \rangle \Rightarrow \mathrm{src}(v_1) \neq \mathrm{src}(v_2)$
3. *For the root w_D of* D*,* $\mathrm{src}(w_D) = w$ *holds.*
4. $q = \mathrm{src}(V_D) \cap R \neq \emptyset$
5. *Each node $v \in V_D$ with $k = idx(\mathrm{src}(v))$ satisfies* at least one *of the two properties (DT1) and (DT2).*

$$\textbf{\textit{(DT1)}} \quad \mathrm{src}(v) \in q \;\wedge\; \mathrm{src}(v) \textit{ is a leaf of } VS^{+}$$
$$\textbf{\textit{(DT2)}} \quad \{\mathrm{src}(b) \mid v \lhd_D b\} \textit{ is } \underline{q_{op}}_k\textit{-minimal}$$

Definition 8 relates a derivation tree D to the underlying EVS by means of a function *src*. This function maps each node $v' \in V_D$ to its *source* node $v = src(v') \in V$. Accordingly, a set of nodes $U = src(U_D) \subseteq V$ with $U_D \subseteq V_D$ is also called *source* of U_D, and

the *source* of D is identified by $src(V_D)$. In the example in Figure 6, the source of D is equal to the set $\{V_1, V_2, V_3, R_1, R_2\}$, and for the sample LDT L in Figure 7 the source of L is equal to the set $\{V_3, R_2\}$. Vice versa, v' is called a *proxy* of v, if $v = src(v')$ holds.

Property (1) ensures that each edge of D has a corresponding edge in its source. Property (2) states that no path in the derivation tree forms a cycle in its source. Note, that the term "$\exists\ \langle v_1, \ldots, v_2 \rangle$" is used to express the fact that there exists a path from v_1 to v_2 in the corresponding graph of the nodes. Such a path can be interpreted as a sequence of recursive invocations within the algorithm execution. Property (3) guarantees that the root of D is a proxy of the root of the corresponding EVS. These first three properties holds for the example in Figure 6, as can be easily deduced by comparing the graph of D with the underlying voting structure in Figure 3. Property (4) enforces that all physical nodes of the source of D (i.e., the *physical source* of D) are exactly the elements of the derived quorum q. Graphically, this means that all visited physical nodes of a derivation are collected in the quorum q. In the example, the physical nodes R_1 and R_2 have been visited in the derivation, which are exactly the elements of the resulting quorum q. Property (5) requires that each node of D is a leaf whose source is part of q *(DT1)* or its direct successors are just sufficient to achieve its scalar operation quorum *(DT2)* (cf. Definition 2). For the sample derivation tree, (DT1) holds for each leaf, and (DT2) holds for every other node $v \in V_D$, since the votes of all direct successors of each node v sum up to the scalar read quorum of v, which also can be seen in the graph of D.

Additionally, we define the concept of an *LDT* to consider derivations locally, starting from a node different than the root.

Definition 9 (Local Derivation Tree)**.** *Let* $\mathrm{VS}^+ = (V, \lhd, w, \underline{v}, \underline{s}, \underline{q_{op}})$ *be an EVS and* $\tilde{q} \subseteq R$ *be a subset of the physical nodes. A tree* $L = (V_L, \lhd_L)$ *with root* w_L *is called* a local derivation tree (LDT) *of* VS^+ *for* $\tilde{q}$ *in operation* op*, if there exists a function* $\mathrm{src} : V_L \to V$ *for which properties (1), (2), (4), and (5) of Definition 8 hold wrt.* L *instead of* D*.*

Note, that Property (3) is *not* required for a LDT.

We now prove that a derivation tree represents a derivation of a *single* quorum, which is an element of the *complete* set of quorums calculated by the algorithm.

Theorem 1 (A Derivation Tree Represents a Quorum)**.** *Let* $\mathrm{VS}^+ = (V, \lhd, w, \underline{v}, \underline{s}, \underline{q_{op}})$ *be an EVS. For each operation* op *and each quorum* $\mathrm{q} \subseteq R$ *the following holds:*

$$q \in quorums^+_{op}(VS^+, w, \emptyset) \Leftrightarrow \exists \text{ derivation tree } D \text{ of } VS^+ \text{ for } q \text{ in } op$$

Since Theorem 1 only relates the results of the initial invocation of the algorithm to corresponding derivation trees, we cannot prove the theorem by induction. Thus, we generalize the relation to any invocation of the algorithm. The idea is to also consider any non-root node a of a derivation tree and the corresponding invocation of $quorums^+_{op}(VS^+, a, Pre)$ with the result Q_a. We relate each of the resulting sets of physical nodes $\tilde{q} \in Q_a$ to a LDT of VS^+ for $\tilde{q}$ in op, starting from node a. To exemplify

this idea, we consider the LDT L of Figure 7, which we relate to the invocation of $quorums^{+}_{op}(VS^{+}, V_3, \{V_1\})$. L starts from $V_3'' = w_L$, the proxy of the invoked node V_3 (i.e., $V_3 = src(w_L)$). Additionally, we can rely on the fact that a node of $Pre = \{V_1\}$ cannot be visited again, i.e., nodes of Pre cannot have proxies in L. L, e.g., does not contain a proxy of V_1 (i.e., $\{V_1\} \cap src(V_L) = \emptyset$.). These considerations lead to the following lemma.

Lemma 1. *Let* $\mathrm{VS}^{+} = (V, \lhd, w, \underline{v}, \underline{s}, \underline{q_{op}})$ *be an EVS. For each operation* op *and each set* $\tilde{q} \subseteq R$*, the following holds:*

$$\begin{aligned} & \tilde{q} \in quorums^{+}_{op}(VS^{+}, a, Pre) \ \wedge \ a \notin \mathrm{Pre} \\ \Leftrightarrow\ & \exists \text{ LDT } L \text{ of } VS^{+} \text{ for } \tilde{q} \text{ in } op \wedge a = src(w_L) \wedge Pre \cap src(V_L) = \emptyset \end{aligned}$$

Proof. The proof is divided into two directions. The proofs for both directions are performed by induction over the structure of VS^{+} wrt. parameter a and uses arguments wrt. the algorithm shown in Figure 4.

$\Rightarrow$:

Base case: Let a be a leaf of VS^{+}. Then the following holds:

$$\begin{aligned} & \tilde{q} \in \mathrm{quorums}^{+}_{op}(\mathrm{VS}^{+}, a, \mathrm{Pre}) \ \wedge \ a \notin Pre \\ \Rightarrow\ & \tilde{q} \in Q_a \wedge (a \in R \Rightarrow Q_a = \{\{a\}\}) \wedge (a \notin R \Rightarrow Q_a = \emptyset) \wedge a \notin Pre \\ \Rightarrow\ & a \in R \wedge \tilde{q} = \{a\} \wedge a \notin Pre \end{aligned}$$

Consider the tree $L = (V_L, \lhd_L) = (\{a'\}, \emptyset)$ with the function src given by $src(a') = a$. L is a LDT of VS^{+} for $\tilde{q}$ in op, since the required properties for a LDT holds (cf. Definition 9):

(1) holds, since $\lhd_L$ is empty.
(2) holds, since $\lhd_L$ is empty.
(4) $\tilde{q} = \{a\} = \{a\} \cap R = src(\{a'\}) \cap R = src(V_L) \cap R$
(5) $src(a') = a \in \tilde{q} \wedge src(a') = a$ is a leaf of VS^{+}

Overall, the following holds:

$$\begin{aligned} & \exists \text{ LDT } L \text{ of } VS^{+} \text{ for } \tilde{q} \text{ in } op \wedge a' = w_L \wedge \{a\} = src(V_L) \wedge a \notin Pre \\ \Rightarrow\ & \exists \text{ LDT } L \text{ of } VS^{+} \text{ for } \tilde{q} \text{ in } op \wedge a = src(w_L) \wedge Pre \cap src(V_L) = \emptyset \end{aligned}$$

Inductive step: Let a be not a leaf of VS^{+}, let nodes $b_1, \ldots, b_l \in V$ be all direct successors of a with $a \lhd b_i \wedge b_i \notin Pre$ for $i \in \{1, \ldots, l\}$, and let $k = \underline{q_{op}}_{idx(a)}$ be the scalar quorum of a for the operation op. Assume $l = 0$ holds, then the following holds:

$$L = \emptyset \Rightarrow Q_a = \emptyset \Rightarrow \tilde{q} \in \emptyset \quad \lightning$$

Thus, $l > 0$ holds and due to Lines 9 – 10, for all $i \in \{1, \ldots, l\}$ the following holds:

$$\begin{aligned} & a \lhd b_i \wedge b_i \notin Pre \wedge Q_{b_i} = quorums^{+}_{op}(VS^{+}, b_i, Pre \cup \{a\}) \\ \Rightarrow\ & a \neq b_i \wedge b_i \notin Pre \wedge Q_{b_i} = quorums^{+}_{op}(VS^{+}, b_i, Pre \cup \{a\}) \\ \Rightarrow\ & b_i \notin \{Pre\} \cup \{a\} \wedge Q_{b_i} = quorums^{+}_{op}(VS^{+}, b_i, Pre \cup \{a\}) \\ \overset{\text{I.H.}}{\Rightarrow}\ & \big(q_i \in Q_{b_i} \Rightarrow \exists \text{ LDT } L_i \text{ for } q_i \text{ in } op \wedge b_i = src(w_{L_i}) \wedge Pre \cap src(V_{L_i}) = \emptyset\big) \end{aligned}$$

Distinguish the two cases $a \notin R$ and $a \in R$. First, let $a \notin R$ be a virtual node of VS^+, then the following holds:

$$\begin{aligned}
L =& \{\bigcup_{i\in I} q_i | \bigwedge_{i\in I} q_i \in Q_{b_i} \wedge \bigcup_{i\in I}\{b_i\} \text{ is } k\text{-minimal} \wedge I \subseteq \{1,\dots,l\}\} \wedge \\
& Q_a = \{m | m \in L\} \wedge \tilde{q} \in Q_a \wedge a \notin Pre \\
\Rightarrow\ & L = \{\dots\} \wedge \tilde{q} \in L \wedge a \notin Pre \\
\Rightarrow\ & \tilde{q} = \bigcup_{i\in I} q_i \wedge \bigwedge_{i\in I} q_i \in Q_{b_i} \wedge \bigcup_{i\in I}\{b_i\} \text{ is } k\text{-minimal } \wedge I \subseteq \{1,\dots,l\} \wedge a \notin Pre \\
\Rightarrow\ & \tilde{q} = \bigcup_{i\in I} q_i \wedge \bigcup_{i\in I}\{b_i\} \text{ is } k\text{-minimal } \wedge I \subseteq \{1,\dots,l\} \wedge a \notin Pre \wedge \\
& \forall i \in I : \big(\exists \text{ LDT } L_i \text{ for } q_i \text{ in } op \wedge b_i = src(w_{L_i}) \wedge (Pre \cup \{a\}) \cap src(V_{L_i}) = \emptyset\big)
\end{aligned}$$

W.l.o.g, assume that for $i \in I$, the set of nodes V_{L_i} are disjoint and $w_{L_i} = b'_i$ holds for the root nodes. Note, that this can be enforced by renaming the nodes. Consider the tree

$$L = (V_L, \lhd_L) = \{\ \{a'\} \cup \bigcup_{i\in I} V_{L_i}, \{a' \lhd_L b'_i | i \in I\} \cup \bigcup_{i\in I} \lhd_{L_i}\}$$

with the function

$$src(v) = \begin{cases} a & \text{if } v = a' \\ src_{L_i}(v) & \text{if } v \in V_{L_i} \end{cases}$$

The tree L is shown schematically in Figure 8. L has the root node $w_L = a'$ and L is a

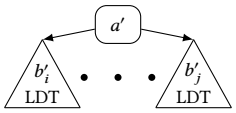

Figure 8: Schematic Representation of LDT L

LDT of VS^+ for $\tilde{q}$ in op, since the required properties for a LDT holds:

(1) Let $v_1, v_2 \in V_L$ be nodes of the LDT L, then the following holds:

$$\begin{aligned}
v_1 \lhd_L v_2 &\Rightarrow (v_1 = a' \wedge v_2 = b'_i \wedge i \in I) \vee (v_1, v_2) \in \bigcup_{i\in I} \lhd_{L_i} \\
&\Rightarrow (src(v_1) = a \wedge src(v_2) = b_i \wedge i \in I) \vee \bigvee_{i\in I} v_1 \lhd_{L_i} v_2 \\
&\Rightarrow src(v_1) \lhd src(v_2)
\end{aligned}$$

(2) Let $v_1, v_2 \in V_L$ be nodes of the LDT L, then the following holds:

$$\begin{aligned}
& \exists\langle v_1, \dots, v_2\rangle \\
\Rightarrow\ & (v_1 = a' \wedge v_2 \in V_{L_i} \wedge i \in I) \vee \exists\langle v_1, \dots, v_2\rangle \text{ in } L_i \\
\Rightarrow\ & (src(v_1) = a \wedge src(v_2) \notin Pre \cup \{a\}) \vee src(v_1) \neq src(v_2) \\
\Rightarrow\ & src(v_1) \neq src(v_2)
\end{aligned}$$

(4) For the result $\tilde{q}$, the following holds:

$$\begin{aligned}\tilde{q} &= \bigcup_{i\in I} q_i = \bigcup_{i\in I}\big(src(V_{L_i}) \cap R\big) = R \cap \bigcup_{i\in I} src(V_{L_i}) = \big(R \cap \bigcup_{i\in I} src(V_{L_i})\big) \cup \emptyset \\ &= \big(R \cap \bigcup_{i\in I} src(V_{L_i})\big) \cup \big(R \cap src(\{a'\})\big) = R \cap \big(\bigcup_{i\in I} src(V_{L_i}) \cup src(\{a'\})\big) \\ &= R \cap src\big(\bigcup_{i\in I} V_{L_i} \cup \{a'\}\big) = src(V_L) \cap R\end{aligned}$$

(5) Let $v \in V_L$ be a node of the LDT L for which $v \in \bigcup_{i\in I} V_{L_i}$ holds. Then *(DT1)* or *(DT2)* holds for v by induction hypothesis. Let $v \in V_L$ be a node of the LDT L for which $v \in \{a'\}$ holds. Then the following holds for v:

$$\begin{aligned}\{src(b)|v \lhd_L b\} &= \{src(b)|a' \lhd_L b\} = \{src(b'_i)|i \in I\} \\ &= \bigcup_{i\in I}\{src(b'_i)\} = \bigcup_{i\in I}\{b_i\}\end{aligned}$$

Thus, *(DT2)* holds for v, since $\{src(b)|v \lhd_L b\} = \bigcup_{i\in I}\{b_i\}$ is k-minimal.

Additionally, the following holds:

$$\begin{aligned}src(w_L) &= src(a') = a \qquad \text{and} \\ Pre \cap src(V_L) &= Pre \cap src\big(\{a'\} \cup \bigcup_{i\in I} V_{L_i}\big) \\ &= Pre \cap \big(src(\{a'\}) \cup \bigcup_{i\in I} src(V_{L_i})\big) \\ &= \big(Pre \cap src(\{a'\})\big) \cup \bigcup_{i\in I}\big(Pre \cap src(V_{L_i})\big) \\ &= \big(Pre \cap \{a\}\big) \cup \bigcup_{i\in I} \emptyset = \emptyset \cup \emptyset = \emptyset\end{aligned}$$

Overall, the following holds:

$$\exists \text{ LDT } L \text{ of } VS^+ \text{ for } \tilde{q} \text{ in } op \wedge a = src(w_L) \wedge Pre \cap src(V_L) = \emptyset$$

Now, let $a \in R$ be a physical node of VS^+, then the following holds:

$$\begin{aligned}L =& \big\{\bigcup_{i\in I} q_i | \bigwedge_{i\in I} q_i \in Q_{b_i} \wedge \bigcup_{i\in I}\{b_i\} \text{ is } k\text{-minimal} \wedge I \subseteq \{1,\dots,l\}\big\} \wedge \\ & Q_a = \{m \cup \{a\} | m \in L\} \wedge \tilde{q} \in Q_a \wedge a \notin Pre \\ \Rightarrow\ & L = \big\{\dots\big\} \wedge \tilde{q} \in \{m \cup \{a\} | m \in L\} \wedge a \notin Pre \\ \Rightarrow\ & \tilde{q} = \big(\bigcup_{i\in I} q_i\big) \cup \{a\} \wedge \bigwedge_{i\in I} q_i \in Q_{b_i} \wedge \bigcup_{i\in I}\{b_i\} \text{ is } k\text{-minimal } \wedge I \subseteq \{1,\dots,l\} \wedge a \notin Pre \\ \Rightarrow\ & \tilde{q} = \{a\} \cup \bigcup_{i\in I} q_i \wedge \bigcup_{i\in I}\{b_i\} \text{ is } k\text{-minimal } \wedge I \subseteq \{1,\dots,l\} \wedge a \notin Pre \wedge \\ & \forall i \in I : \big(\exists \text{ LDT } L_i \text{ for } q_i \text{ in } op \wedge b_i = src(w_{L_i}) \wedge (Pre \cup \{a\}) \cap src(V_{L_i}) = \emptyset\big)\end{aligned}$$

The proof from this point on is analogous to case $a \notin R$, except for the verification of Property (4) of the considered LDT L. The adapted verification of Property (4) is given in the following:

$$\begin{aligned}\tilde{q} &= \{a\} \cup \bigcup_{i \in I} q_i = \{a\} \cup \bigcup_{i \in I} \big(src(V_{L_i}) \cap R\big) = (\{a\} \cap R) \cup \bigcup_{i \in I} \big(src(V_{L_i}) \cap R\big) \\ &= \big(\{a\} \cup \bigcup_{i \in I} src(V_{L_i})\big) \cap R = \big(src(\{a'\}) \cup \bigcup_{i \in I} src(V_{L_i})\big) \cap R \\ &= src\big(\{a'\} \cup \bigcup_{i \in I} V_{L_i}\big) \cap R = src(V_L) \cap R\end{aligned}$$

□

$\Leftarrow$:

Base case: Let a be a leaf of VS^+. Then $quorums^+_{op}(VS^+, a, Pre) = \{a\}$ holds if $a \in R$; otherwise $quorums^+_{op}(VS^+, a, Pre) = \emptyset$ holds as can be easily read in the definition of the algorithm. Consider a LDT L of VS^+ for $\tilde{q}$ in op for which $a = src(w_L) \wedge Pre \cap src(V_L) = \emptyset$ holds. Such a LDT L exists by assumption. Assume $a' = w_L$ is not a leaf of L. Then the following holds:

$$\begin{aligned}& a' \text{ is not a leaf of } L \\ \Rightarrow\ & \exists b' \in V_L : a' \lhd_L b' \\ \Rightarrow\ & src(a') \lhd src(b') \\ \Rightarrow\ & src(a') = a \text{ is not a leaf of } VS^+ \quad \lightning\end{aligned}$$

Therefore, a' is a leaf of L. Since a' is also the root, it is the only node of L. Thus, the following holds:

$$\begin{aligned}& V_L = \{a'\} \wedge \tilde{q} = src(V_L) \cap R \wedge Pre \cap src(V_L) = \emptyset \\ \Rightarrow\ & src(V_L) = \{a\} \wedge \tilde{q} = src(V_L) \cap R \wedge Pre \cap src(V_L) = \emptyset \\ \Rightarrow\ & \tilde{q} = \{a\} \cap R \wedge Pre \cap \{a\} = \emptyset \\ \Rightarrow\ & (a \in R \Rightarrow \tilde{q} = \{a\}) \wedge (a \notin R \Rightarrow \tilde{q} = \emptyset) \wedge a \notin Pre \\ \Rightarrow\ & \tilde{q} = quorums^+_{op}(VS^+, a, Pre) \wedge a \notin Pre\end{aligned}$$

Inductive step: Let a be not a leaf of VS^+. Consider a LDT L of VS^+ for $\tilde{q}$ in op for which $a = src(w_L) \wedge Pre \cap src(V_L) = \emptyset$ holds. Such a LDT L exists by assumption. Since *(DT1)* cannot be true for the root node $a' = w_L$, which is not a leaf, *(DT2)* must hold for a'. Thus, the LDT has the structure shown in Figure 8. Let I be a suitable index set for the given *LDT*, s.t. for each $i \in I$ exists a subtree L_i of a' with the root b'_i, which is a proxy of b_i. First, we show that each subtree L_i of a' is a *LDT* for a set of physical nodes q_i in op with the function $src_{L_i}(v) := src(v)$ for each $v \in V_{L_i}$[3] and for which $b_i = src(w_{L_i}) \wedge (Pre \cup \{a\}) \cap src(V_{L_i}) = \emptyset$ holds.

[3] src_{L_i} is identical to src on the domain of src_{L_i}. Therefore, we only use src in the remainder of the proof.

(1) Let $v_1, v_2 \in V_L$ be nodes of the subtree L_i, then the following holds:

$$v_1 \lhd_{L_i} v_2 \Rightarrow v_1 \lhd_L v_2 \Rightarrow \mathit{src}(v_1) \lhd \mathit{src}(v_2)$$

(2) Let $v_1, v_2 \in V_L$ be nodes of the subtree L_i, then the following holds:

$$\exists\langle v_1, \ldots, v_2\rangle \text{ in } L_i \Rightarrow \exists\langle v_1, \ldots, v_2\rangle \text{ in } L \Rightarrow \mathit{src}(v_1) \neq \mathit{src}(v_2)$$

(4) $q_i = \mathit{src}(V_{L_i}) \cap R$ is well-defined.

(5) Let $v \in V_{L_i}$ be a node of the subtree L_i and let $k = \underline{q_{op}}_{idx(\mathit{src}(v))}$ be the scalar quorum of $\mathit{src}(v)$ for the operation op. Then the following holds for v:

$$\begin{aligned}
& v \in V_{L_i} \\
\Rightarrow\ & \mathit{src}(v) \in \mathit{src}(V_{L_i}) \wedge v \in V_L \\
\Rightarrow\ & \mathit{src}(v) \in \mathit{src}(V_{L_i}) \wedge \big((\mathit{src}(v) \in \tilde{q} \wedge \mathit{src}(v) \text{ is a leaf of } VS^+) \vee \\
& \qquad \{\mathit{src}(b) | v \lhd_L b\} \text{ is k-minimal } \big) \\
\Rightarrow\ & \big(\mathit{src}(v) \in \mathit{src}(V_{L_i}) \wedge \mathit{src}(v) \in R \wedge \mathit{src}(v) \text{ is a leaf of } VS^+\big) \vee \\
& \qquad \{\mathit{src}(b) | v \lhd_{L_i} b\} \text{ is k-minimal} \\
\Rightarrow\ & \big(\mathit{src}(v) \in \mathit{src}(V_{L_i}) \cap R = q_i \wedge \mathit{src}(v) \text{ is a leaf of } VS^+\big) \vee \\
& \qquad \{\mathit{src}(b) | v \lhd_{L_i} b\} \text{ is k-minimal} \\
\Rightarrow\ & (DT1) \vee (DT2)
\end{aligned}$$

Additionally, the following holds:

$$\begin{aligned}
& \forall v \in V_{L_i} : \exists\langle a', \ldots, v\rangle \\
\Rightarrow\ & \forall v \in V_{L_i} : \mathit{src}(a') \neq \mathit{src}(v) \\
\Rightarrow\ & \mathit{src}(a') \notin \mathit{src}(V_{L_i}) \\
\Rightarrow\ & a \notin \mathit{src}(V_{L_i})
\end{aligned}$$

Therefore, the following holds for the LDT L:

$$\begin{aligned}
\mathit{src}(w_{L_i}) = \mathit{src}(b_i') = b_i \qquad & \text{and} \\
(\mathit{Pre} \cup \{a\}) \cap \mathit{src}(V_{L_i}) = \underbrace{(\mathit{Pre} \cap \mathit{src}(V_{L_i}))}_{=\emptyset} \cup \underbrace{(\{a\} \cap \mathit{src}(V_{L_i}))}_{=\emptyset} = \emptyset
\end{aligned}$$

Since each subtree L_i is a LDT of VS^+ for q_i in op for which $b_i = \mathit{src}(w_{L_i}) \wedge (\mathit{Pre} \cup \{a\}) \cap \mathit{src}(V_{L_i}) = \emptyset$ holds and due to the induction hypothesis, $q_i \in \mathit{quorums}^+_{op}(VS^+, b_i, \mathit{Pre} \cup$

$\{a\}$) holds . Furthermore, the following holds:

$$\begin{aligned}
\tilde{q} &= \mathit{src}(V_L) \cap R \\
&= \mathit{src}(\{a'\} \cup \bigcup_{i \in I} V_{L_i}) \cap R \\
&= \big(\mathit{src}(\{a'\}) \cup \bigcup_{i \in I} \mathit{src}(V_{L_i})\big) \cap R \\
&= \big(\mathit{src}(\{a'\}) \cap R)\big) \cup \bigcup_{i \in I} \big(\mathit{src}(V_{L_i}) \cap R\big) \\
&= (\{a\} \cap R) \cup \bigcup_{i \in I} q_i \\
\Rightarrow & \big(a \in R \Rightarrow \tilde{q} = \{a\} \cup \bigcup_{i \in I} q_i\big) \wedge \big(a \notin R \Rightarrow \tilde{q} = \bigcup_{i \in I} q_i\big)
\end{aligned}$$

Finally, the following holds:

$$\begin{aligned}
& \bigcup_{i \in I} q_i \in L \wedge \mathit{Pre} \cap \mathit{src}(V_L) = \emptyset \wedge a = \mathit{src}(w_L) \\
\Rightarrow\ & \big(a \in R \Rightarrow \{a\} \cup \bigcup_{i \in I} q_i \in Q_a\big) \wedge \big(a \notin R \Rightarrow \bigcup_{i \in I} q_i \in Q_a\big) \wedge \\
& \quad \mathit{Pre} \cap \mathit{src}(V_L) = \emptyset \wedge a \in \mathit{src}(V_L) \\
\Rightarrow\ & (a \in R \Rightarrow \tilde{q} \in Q_a) \wedge (a \notin R \Rightarrow \tilde{q} Q_a) \wedge a \notin \mathit{Pre} \\
\Rightarrow\ & \tilde{q} \in \mathit{quorums}^+_{op}(VS^+, a, \mathit{Pre}) \wedge (a \notin R \Rightarrow \tilde{q} \in Q_a) \wedge a \notin \mathit{Pre}
\end{aligned}$$

□

This concludes the overall proof. □

With Lemma 1, we have a property for each invocation of the algorithm that we can use in order to prove the correctness of Theorem 1. This theorem is concerned with the initial invocation.

Proof of Theorem 1. Let $VS^+ = (V, \lhd, w, \underline{v}, \underline{s}, \underline{q_{op}})$ be an EVS. Consider the initial invocation of $\mathit{quorums}^+_{op}(VS^+, a, \mathit{Pre})$ with $a = w$, $\mathit{Pre} = \emptyset$:

$$\begin{aligned}
& q \in \ \mathit{quorums}^+_{op}(VS^+, w, \emptyset) \ \wedge \ w \notin \emptyset \\
\overset{1}{\Leftrightarrow}\ & \exists \text{ LDT } L \text{ of } VS^+ \text{ for } q \text{ in } op \wedge w = \mathrm{src}(w_L) \wedge \emptyset \cap \mathrm{src}(V_L) = \emptyset \\
\overset{(*)}{\Leftrightarrow}\ & \exists \text{ derivation tree } D \text{ of } VS^+ \text{ for } q \text{ in } op
\end{aligned}$$

The terms $w \notin \emptyset$ and $\emptyset \cap \mathit{src}(V_L) = \emptyset$ are tautologies and, therefore, have no influence on the evaluation of the conjunctions. For deducing the equivalence "$(*)$," we use Definitions 8 and 9. The LDT becomes a derivation tree, since $w = \mathrm{src}(w_L)$ is property (3) of a derivation tree and the remaining properties hold for a LDT by definition. □

6 Formal Proofs

6.1 Backward Compatibility

As a prerequisite for the formal proof of backward compatibility, we establish an invariant for recursive invocations to $quorums^{+}_{op}$. This invariant formalizes the condition that node a has not been visited by any recursive step leading to the invocation under consideration and that set *Pre* contains all visited nodes of the recursive steps (i.e., there exists a path from each node in set *Pre* to the node a in VS^{+}). The idea of an invariant in recursive algorithms is that its property holds in each recursion step. The precondition for this is that the invariant is already valid for the current invocation, i.e., in particular, the initial invocation must satisfy the invariant.

Lemma 2 (Invariant for the New Algorithm)**.** *The property* (Inv) $a \notin Pre \ \wedge \ \forall v \in Pre : \ \exists \ \langle v, \ldots, a \rangle$ *is an invariant of the recursive algorithm* $\text{quorums}^{+}_{\text{op}}$(in: VS^{+}, a, Pre).

Proof. Consider any recursive invocation of $quorums^{+}_{op}$ that is executed by the new algorithm through the invocation of $quorums^{+}_{op}(VS^{+},\ a,\ Pre)$ for which the invariant *(Inv)* holds. In Line 10, the following holds for the recursive invocation of $quorums^{+}_{op}(VS^{+},\ b_i,\ Pre \cup \{a\})$:

$$
\begin{aligned}
& \overbrace{b_i \notin Pre \wedge a \lhd b_i}^{\textit{Line 9}} \wedge \overbrace{a \notin Pre \ \wedge \ \forall v \in Pre : \ \exists \ \langle v, \ldots, a \rangle}^{\textit{(Inv)}} \\
\Rightarrow \quad & b_i \notin Pre \wedge \exists \langle a, b_i \rangle \wedge \forall v \in Pre : \exists \langle v, \ldots, a, b_i \rangle \\
\Rightarrow \quad & b_i \notin Pre \wedge \forall v \in Pre \cup \{a\} : \exists \langle v, \ldots, b_i \rangle
\end{aligned}
$$

In addition, for the initial invocation with $Pre = \emptyset$, the following holds:

$$a \notin \emptyset \wedge \forall v \in \emptyset : \exists \langle v, \ldots, a \rangle$$

□

Now, that we have shown the validity of the invariant for the new algorithm, the proof of backward compatibility follows.

Theorem 2 (Backward Compatibility)**.** *The new algorithm* $\text{quorums}^{+}_{\text{op}}$ *calculates for each acyclic voting structure* VS^{+} *the same results as the old algorithm.*

Proof. Let VS^{+} be an acyclic voting structure, $a \in VS^{+}$ a node of VS^{+}, and $Pre \subseteq V$ a subset of the nodes of VS^{+}, for which *(Inv)* holds. We consider the invocation of $quorums^{+}_{op}(VS^{+}, a, Pre)$ and compare the results of the execution of the new and the old[4] algorithm by an inductive argument over the structure of VS^{+} wrt. parameter a.

Base case: Let node a be a leaf. Then, Lines 2–7 and Line 18 are executed by both the new and the old algorithm. For these lines, both algorithms are identical as are their results.

[4]The old algorithm can be derived by the new algorithm by removing the parameter *Pre* and all its usages, $b_i \notin Pre$ in Line 9 and $Pre \cup \{a\}$ in Line 10.

Inductive step: Let node a be *not* a leaf. Then, Lines 2–3 and Lines 9–18 are executed by both the new and the old algorithm. Assume that $a \lhd b_i \wedge b_i \in Pre$ holds for any node b_i. Then, the following holds:

$$\begin{aligned} & \overbrace{a \lhd b_i \wedge b_i \in Pre}^{\text{Assumption}} \wedge \overbrace{a \notin \text{Pre} \wedge \forall v \in \text{Pre} : \exists \langle v, \ldots, a \rangle}^{(Inv)} \\ \Rightarrow \quad & a \lhd b_i \wedge \exists \langle b_i, \ldots, a \rangle \\ \Rightarrow \quad & \exists \langle b_i, \ldots, b_i \rangle \quad \lightning \quad VS^+ \text{ is acyclic} \end{aligned}$$

Thus, $b_i \notin Pre$ holds for all nodes $b_i, i = 1, \ldots, l$ with $a \lhd b_i$. Hence, the new algorithm iterates over the same nodes b_i as the old algorithm in Line 9. Therefore, recursive invocations are executed for the same nodes b_i in Line 10 and their results Q_{b_i} are identical for the new and old algorithm by induction hypothesis. Having identical results Q_{b_i}, both algorithms calculate the same results Q_a in Lines 11–18. □

6.2 Termination

With a decreasing function and the invariant *(Inv)* of Lemma 2, we can formally prove that the algorithm terminates.

Theorem 3 (Termination)**.** *The new algorithm* $\text{quorums}^+_{\text{op}}$ *terminates for each invocation with every EVS* VS^+*, any node* $\text{a} \in V$*, and any set* $\text{Pre} \subseteq V$ *under the precondition* (Inv) *after at most* $|V| - |\text{Pre}|$ *recursive invocations.*

Proof. First, we define the decreasing function for the algorithm $quorums^+_{op}(VS^+, a, Pre)$ as follows:

$$h : U_{VS^+} \times V \times \mathbb{P}(V) \to \mathbb{N}_0, \qquad h(VS^+, a, Pre) := |V| - |Pre| \geq 0$$

Consider the recursive invocation of $quorums^+_{op}(VS^+, b_i, Pre \cup \{a\})$ in Line 10. Since $a \notin Pre$ holds due to *(Inv)*, the following also holds:

$$h(VS^+, b_i, Pre \cup \{a\}) = |V| - |Pre \cup \{a\}| = |V| - |Pre| - 1 = h(VS^+, a, Pre) - 1$$

Thus, h decreases by one with each recursive step and cannot get any lower than *zero*. We can conclude that the invocation under consideration terminates after at most $|V| - |Pre|$ recursive steps. □

Since the precondition *(Inv)* of Theorem 3 holds for the initial invocation, we conclude the termination of the algorithm after at most $|V|$ recursive steps.

6.3 Correctness of the Design Pattern

In [11] we have presented following theorem:

Theorem 4 (Equivalence [11])**.** *Let $GS := (R, \{\square, \diamond\}, S_\square, S_\diamond, E_\square, E_\diamond)$ be a grid structure, $Q_R, Q_W \subseteq \mathbb{P}(R)$ be the derived sets of read and write quorums, and $VS^+ = VSP(GS)$ be the modeled EVS by the design pattern for path-based replication strategies. Then, the following holds:*

$$Q_R = quorums^+_{read}(VS^+, w, \emptyset) \;\; \wedge$$
$$Q_W = quorums^+_{write}(VS^+, w, \emptyset)$$

The theorem establishes the connection between EVS and PBRS wrt. the introduced design pattern. In this section, we present the corresponding *formal* proof of Theorem 4, which uses the concept of derivation trees introduced in Section 5. As a preparation, we introduce two definitions and a lemma, which will be used in the proof.

Definition 10. *Let $R = \{R_1, \ldots, R_n\}$ be the set of nodes of a distributed system, let* GS *be a grid structure, let* $\mathrm{VS}^+ = \mathrm{VSP(GS)}$ *be the corresponding EVS, and let* D *be a derivation tree of* VS^+ *for a quorum* q *in operation* op*. The predicate $P_V(k_1, \ldots, k_m)$ is defined as follows:*

$$\begin{aligned}
&P_V(k_1, \ldots, k_m) :\Leftrightarrow \\
&1 \leq m \leq n \wedge \{k_1, \ldots, k_m\} \subseteq \{1, \ldots, n\} \wedge |\{k_1, \ldots, k_m\}| = m \wedge R_{k_1} \in S_\diamond \;\wedge \\
&\forall i \in \{1, \ldots, m\} : \mathrm{src}(V'_{k_i}) = V_{k_i} \;\wedge \\
&\forall i \in \{1, \ldots, m-1\} : \big((R_{k_i}, R_{k_{i+1}}) \in \diamond \wedge \exists \langle V'_{k_i}, \ldots, V'_{k_m} \rangle\big)
\end{aligned}$$

The predicate $P_H(k_1, \ldots, k_m)$ is defined analogous by replacing each V *by an* H *and each $\diamond$ by a $\square$.*

Definition 11. *Let* VS^+ *be an EVS, let $v \in V$ be a node of* VS^+*, and let $k := \mathrm{idx}(v)$ be the index of* v *in $\underline{v}$. Then*

$$\mathrm{MS_{op}}(v) := \begin{cases} \emptyset & \textit{if } \mathrm{v} \textit{ is a leaf} \\ \{U \subseteq \{b | v \lhd b\} | U \textit{ is } \underline{q_{\mathrm{op}}}_k\textit{–minimal}\} & \textit{otherwise} \end{cases}$$

is the set of the so-called minimal successor sets *of node* v *for operation* op*.* $\mathrm{MS_{op}}(v)$ *contains all possible combinations of successors of* v *whose votes are just sufficient to achieve the scalar operation quorum of* v*.*

Lemma 3. *Let* VS^+ *be an EVS, let* D *be a derivation tree of* VS^+ *for a quorum* q *in operation* op*, let $v \in V_D$ be a node of* D *for which* $\mathrm{src}(v)$ *is not a leaf of* VS^+*, and let $k := \mathrm{idx}(\mathrm{src}(v))$ be the index of the source of* v *in $\underline{v}$. Then the following holds:*

$$\exists U \in \mathrm{MS_{op}}(\mathrm{src}(v)) : \forall u \in U : \exists v' \in V_D : \mathrm{src}(v') = u \wedge v \lhd_D v'$$

Proof.

$$
\begin{aligned}
& v \in V_D \wedge src(v) \text{ is not a leaf of } VS^+ \\
\Rightarrow & \{src(b) | v \lhd_D b\} \text{ is } \underline{q_{op}}_k\text{–minimal} \wedge \forall b \in V_D : \big(v \lhd_D b \Rightarrow src(v) \lhd src(b)\big) \\
\Rightarrow & \{src(b) | v \lhd_D b\} \text{ is } \underline{q_{op}}_k\text{–minimal} \wedge \{src(b) | v \lhd_D b\} \subseteq \{b | src(v) \lhd b\} \\
\Rightarrow & \{src(b) | v \lhd_D b\} \in MS_{op}(src(v)) \\
\Rightarrow & \{src(b) | v \lhd_D b\} \in MS_{op}(src(v)) \wedge \forall u \in \{src(b) | v \lhd_D b\} : \exists b \in V_D : src(b) = u \wedge v \lhd_D b \\
\Rightarrow & \exists U MS_{op}(src(v)) : \forall u \in U : \exists v' \in V_D : src(v') = u \wedge v \lhd_D v'
\end{aligned}
$$

□

Now we continue with the proof of Theorem 4.

Proof for Operation Read. We show that the equivalence $Q_R = quorums^+_{read}(VS^+, w, \emptyset)$ for read quorums holds, by showing two subset relations separately.

$\subseteq$:

Let $q \in Q_R$ be a read quorum of the grid structure GS. Then, q contains the nodes of a vertical path *or* a horizontal path. Let q contain a *vertical* path (the proof is analogous for a *horizontal* path). Then, the following holds:

$$
\begin{aligned}
& q = \{p_1, \ldots, p_m\} \wedge \forall i \in \{1, \ldots, m\} : p_i \in R \wedge \\
& \forall i \in \{1, \ldots, m-1\} : (p_i, p_{i+1}) \in \diamond \wedge p_1 \in S_\diamond \wedge p_m \in E_\diamond
\end{aligned}
$$

Additionally, we assume w.l.o.g.:

$$\forall i \neq 1 : p_i \notin S_\diamond \wedge \forall i \neq m : p_i \notin E_\diamond$$

Now consider the tree $D = (V_D, \lhd_D)$ with

$$
\begin{aligned}
V_D =& \{G', V', v'_1, \ldots, v'_m, p'_1, \ldots, p'_m\} \\
\lhd_D =& \{(G', V'), (V', v'_1)\} \cup \{(v'_i, v'_{i+1}) | 1 \leq i \leq m-1\} \cup \\
& \{(v'_i, p'_i) | 1 \leq i \leq m\}
\end{aligned}
$$

for which we show that it is a derivation tree, by means of the function *src* defined as follows:

$$
\begin{aligned}
src(G') &:= G \\
src(V') &:= V^0 \\
src(v'_i) &:= V_{map(i)} \quad \forall i \in \{1, \ldots, m\} \\
src(p'_i) &:= R_{map(i)} \quad \forall i \in \{1, \ldots, m\}
\end{aligned}
$$

$$
\begin{aligned}
\text{where } map &: \{1, \ldots, m\} \to \{1, \ldots, n\} \\
map(i) = j &:\Leftrightarrow p_i = R_j
\end{aligned}
$$

Note, that *map* is injective. Thus, *src* is obviously an injective function. Now we show that D is a derivation tree of VS^+ for q in operation *read*:

(1) Let $v_1, v_2 \in V_D$ be nodes of D with $v_1 \lhd_D v_2$:

$$G' \lhd_D V' : G \lhd V^0 \Rightarrow \mathit{src}(G') \lhd \mathit{src}(V')$$
$$V' \lhd_D v'_1 : p_1 \in S_\diamond \Rightarrow R_{map(1)} \in S_\diamond \Rightarrow V^0 \lhd V_{map(1)} \Rightarrow \mathit{src}(V') \lhd \mathit{src}(v'_1)$$
$$\forall i \in \{1,\ldots,m-1\}: \quad v'_i \lhd_D v'_{i+1} : (p_i, p_{i+1}) \in \diamond \Rightarrow (R_{map(i)}, R_{map(i+1)}) \in \diamond \Rightarrow V_{map(i)} \lhd V_{map(i+1)} \Rightarrow \mathit{src}(v'_i) \lhd \mathit{src}(v'_{i+1})$$
$$\forall i \in \{1,\ldots,m\}: \quad v'_i \lhd_D p'_i : 1 \leq i \leq m \Rightarrow 1 \leq \mathit{map}(i) \leq n \Rightarrow V_{map(i)} \lhd R_{map(i)} \Rightarrow \mathit{src}(v'_i) \lhd \mathit{src}(p'_i)$$

(2) Let $v_1, v_2 \in V_D$ be nodes of D with $\exists\langle v_1, \ldots, v_2\rangle$:

$$\exists\langle v_1, \ldots, v_2\rangle \Rightarrow v_1 \neq v_2 \Rightarrow \mathit{src}(v_1) \neq \mathit{src}(v_2)$$

(3) G' is the root w_D of D, since G' is the only node without predecessor. Therefore, $\mathit{src}(w_D) = \mathit{src}(G') = G = w$ holds.

(4) For the read quorum $q = \{p_1, \ldots, p_m\}$, the following holds:

$$\begin{aligned}
q &= \{R_{map(i)} | i \in \{1,\ldots,m\}\} = \{R_{map(i)} | i \in \{1,\ldots,m\}\} \cap R \\
&= \emptyset \cup \emptyset \cup (\{R_{map(i)} | i \in \{1,\ldots,m\}\} \cap R) \\
&= (\{G, V^0\} \cap R) \cup (\{V_{map(i)} | i \in \{1,\ldots,m\}\} \cap R) \cup (\{R_{map(i)} | i \in \{1,\ldots,m\}\} \cap R) \\
&= (\{G, V^0\} \cup \{V_{map(i)} | i \in \{1,\ldots,m\}\} \cup \{R_{map(i)} | i \in \{1,\ldots,m\}\}) \cap R \\
&= \mathit{src}(\{G', V', v'_1, \ldots, v'_m, p'_1, \ldots, p'_m\}) \cap R \\
&= \mathit{src}(V_D) \cap R
\end{aligned}$$

(5) Let $v \in V_D$ be a node of D and let $k = \mathit{idx}(\mathit{src}(v))$ denote the index of the source of v in the sequence $\underline{v}$ of VS^+:

$$\begin{aligned}
v = G' &: \{\mathit{src}(b) | G' \lhd_D b\} = \{\mathit{src}(V')\} = \{V^0\} \text{ is } \mathit{1}\text{-minimal} \wedge \underline{q_{read}}_k = 1 \\
v = V' &: \{\mathit{src}(b) | V' \lhd_D b\} = \{\mathit{src}(v'_1)\} = \{V_{map(1)}\} \text{ is } \mathit{1}\text{-minimal} \wedge \underline{q_{read}}_k = 1 \\
i \in \{1,\ldots,m-1\} \quad v = v'_i &: \{\mathit{src}(b) | v'_i \lhd_D b\} = \{\mathit{src}(v'_{i+1}), \mathit{src}(p'_i))\} = \{V_{map(i+1)}, R_{map(i)}\} \\
&\quad \text{is } n{+}\mathit{1}\text{-minimal} \wedge \underline{q_{read}}_k = n+1 \\
v = v'_m &: \{\mathit{src}(b) | v'_m \lhd_D b\} = \{\mathit{src}(p'_m)\} = \{R_{map(m)}\} \text{ is } n\text{-minimal} \wedge \underline{q_{read}}_k = n \\
i \in \{1,\ldots,m\} \quad v = p'_i &: \mathit{src}(p'_i) = R_{map(i)} = p_i \in q \wedge \mathit{src}(p'_i) = R_{map(i)} \text{ is a leaf of } VS^+
\end{aligned}$$

In total, $\exists$ derivation tree D of VS^+ for q in *read* holds. Thus, due to Theorem 1, $q \in \mathit{quorums}^+_{read}(VS^+, w, \emptyset)$ holds. $\square$

$\supseteq$:

Let $q \in \mathit{quorums}^+_{read}(VS^+, w, \emptyset)$ be a read quorum of VS^+. Due to Theorem 1, $\exists$ derivation tree D of VS^+ for q in *read*. We prove the following three statements:

(1) $\exists k_1 \in \{1, \ldots, n\} : \big(P_V(k_1) \vee P_H(k_1)\big)$

(2) $P_V(k_1, \ldots, k_m) \Rightarrow \big(R_{k_m} \in E_\diamond \vee \exists k_{m+1} : P_V(k_1, \ldots, k_{m+1})\big) \quad \wedge$
$P_H(k_1, \ldots, k_m) \Rightarrow \big(R_{k_m} \in E_\square \vee \exists k_{m+1} : P_H(k_1, \ldots, k_{m+1})\big)$

(3) $P_V(k_1, \ldots, k_m) \Rightarrow \forall i \in \{1, \ldots, m\} : R_{k_i} \in q \quad \wedge$
$P_H(k_1, \ldots, k_m) \Rightarrow \forall i \in \{1, \ldots, m\} : R_{k_i} \in q$

Under the assumption that (1), (2), and (3) hold, the following holds:

$$\begin{aligned}
&\exists\ k_1, \ldots, k_m \in \{1, \ldots, n\} : \\
&\quad \big(P_V(k_1, \ldots, k_m) \wedge R_{k_m} \in E_\diamond \wedge \forall i \in \{1, \ldots, m\} : R_{k_i} \in q\big) \vee \\
&\quad \big(P_H(k_1, \ldots, k_m) \wedge R_{k_m} \in E_\square \wedge \forall i \in \{1, \ldots, m\} : R_{k_i} \in q\big) \\
&\Rightarrow \big(R_{k_1} \in S_\diamond \wedge \forall i \in \{1, \ldots, m-1\} : (R_{k_i}, R_{k_{i+1}}) \in \diamond \wedge R_{k_m} \in E_\diamond \wedge \\
&\qquad \forall i \in \{1, \ldots, m\} : R_{k_i} \in q\big) \vee \\
&\quad \big(R_{k_1} \in S_\square \wedge \forall i \in \{1, \ldots, m-1\} : (R_{k_i}, R_{k_{i+1}}) \in \square \wedge R_{k_m} \in E_\square \wedge \\
&\qquad \forall i \in \{1, \ldots, m\} : R_{k_i} \in q\big) \\
&\Rightarrow q \text{ contains a vertical path in } GS \qquad \vee \\
&\quad q \text{ contains a horizontal path in } GS \\
&\Rightarrow q \in Q_R
\end{aligned}$$

Thus, it is left to prove the statements (1), (2), and (3):

(1)

$$\begin{aligned}
&G' = w_D \in V_D \wedge \mathit{src}(w_D) = w = G \\
&\Rightarrow G' \in V_D \wedge \mathit{src}(G') = G \text{ is not a leaf of } VS^+ \\
&\Rightarrow \exists U \in \big\{\{V^0\}, \{H^0\}\big\} : \forall u \in U : \exists v' \in V_D : \mathit{src}(v') = u \\
&\Rightarrow \exists v' \in V_D : \mathit{src}(v') = V^0 \vee \exists v' \in V_D : \mathit{src}(v') = H^0
\end{aligned}$$

Consider the case $\exists v' \in V_D : \mathit{src}(v') = V^0$ and let $V' \in V_D$ be such a v', then the following holds:

$$\begin{aligned}
&V' \in V_D \wedge \mathit{src}(V') = V^0 \text{ is not a leaf of } VS^+ \\
&\Rightarrow \exists U \in \big\{\{V_k\} | R_k \in S_\diamond\big\} : \forall u \in U : \exists v' \in V_D : \mathit{src}(v') = u \\
&\Rightarrow \exists k \in \{1, \ldots, n\} : (R_k \in S_\diamond \wedge \exists v' \in V_D : \mathit{src}(v') = V_k)
\end{aligned}$$

Let $k_1 \in \{1, \ldots, n\}$ be such a k and let V'_{k_1} be the corresponding v', then the following holds:

$$\begin{aligned}
&k_1 \in \{1, \ldots, n\} \wedge R_{k_1} \in S_\diamond \wedge \mathit{src}(V'_{k_1}) = V_{k_1} \\
&\Rightarrow 1 \leq 1 \leq n \wedge \{k_1\} \subseteq \{1, \ldots, n\} \wedge |\{k_1\}| = 1 \wedge R_{k_1} \in S_\diamond \wedge \\
&\quad \forall i \in \{1\} : \mathit{src}(V'_{k_i}) = V_{k_i} \wedge \\
&\quad \forall i \in \{1, \ldots, 1-1\} = \emptyset : \big((R_{k_i}, R_{k_{i+1}}) \in \diamond \wedge \exists \langle V'_{k_i}, \ldots, V'_{k_{i+1}} \rangle\big) \\
&\Rightarrow P_V(k_1)
\end{aligned}$$

The arguments for $\exists v' \in V_D : src(v') = H^0$ implies $P_H(k_1)$ are analogous. Overall $\exists k_1 \in \{1, \dots, n\} : \big(P_V(k_1) \vee P_H(k_1)\big)$ holds.

(2) Assume that $P_V(k_1, \dots, k_m)$, $R_{k_m} \in E_\diamond$ hold, then $R_{k_m} \in E_\diamond \vee \exists k_{m+1} : P_V(k_1, \dots, k_{m+1})$ holds obviously. Now assume that $P_V(k_1, \dots, k_m)$ and $R_{k_m} \notin E_\diamond$ hold, then the following holds:

$$\begin{aligned} & V'_{k_m} \in V_D \wedge src(V'_{k_m}) = V_{k_m} \text{ is not a leaf of } VS^+ \wedge R_{k_m} \notin E_\diamond \\ \Rightarrow\ & \exists U \in \big\{\{V_j, R_{k_m}\} | (R_{k_m}, R_j) \in \diamond\big\} : \forall u \in U : \exists v' \in V_D : src(v') = u \wedge V'_{k_m} \lhd_D v' \end{aligned}$$

Let $k_{m+1} \in \{1, \dots, n\}$ be a suitable j for such a U and let $V'_{k_{m+1}}$ be a corresponding v', then the following holds:

$$\begin{aligned} & k_{m+1} \in \{1, \dots, n\} \wedge (R_{k_m}, R_{k_{m+1}}) \in \diamond \wedge src(V'_{k_{m+1}}) = V_{k_{m+1}} \wedge V'_{k_m} \lhd_D V'_{k_{m+1}} \\ \Rightarrow\ & k_{m+1} \in \{1, \dots, n\} \wedge \{k_1, \dots, k_m\} \subseteq \{1, \dots, n\} \wedge R_{k_1} \in S_\diamond \wedge \\ & \quad src(V'_{k_{m+1}}) = V_{k_{m+1}} \wedge \forall i \in \{1, \dots, m\} : src(V'_{k_i}) = V_{k_i} \wedge \\ & \quad (R_{k_m}, R_{k_{m+1}}) \in \diamond \wedge \forall i \in \{1, \dots, m-1\} : (R_{k_i}, R_{k_{i+1}}) \in \diamond \wedge \\ & \quad V'_{k_m} \lhd_D V'_{k_{m+1}} \wedge \forall i \in \{1, \dots, m-1\} : \exists \langle V'_{k_i}, \dots, V'_{k_m} \rangle \\ \Rightarrow\ & \{k_1, \dots, k_{m+1}\} \subseteq \{1, \dots, n\} \wedge R_{k_1} \in S_\diamond \wedge \\ & \quad \forall i \in \{1, \dots, m+1\} : src(V'_{k_i}) = V_{k_i} \wedge \\ & \quad \forall i \in \{1, \dots, m\} : (R_{k_i}, R_{k_{i+1}}) \in \diamond \wedge \\ & \quad \forall i \in \{1, \dots, m\} : \exists \langle V'_{k_i}, \dots, V'_{k_{m+1}} \rangle \end{aligned}$$

Since $\forall i \in \{1, \dots, m\} : \exists \langle V'_{k_i}, \dots, V'_{k_{m+1}} \rangle$ holds and due to Property (2) of Definition 8, the following holds for each $i \in \{1, \dots, m\}$:

$$V_{k_i} \neq V_{k_{m+1}} \Rightarrow k_i \neq k_{m+1}$$

Thus, $|\{k_1, \dots, k_{m+1}\}| = m + 1$ holds and this implies $1 \leq m + 1 \leq n$. In total, the following holds:

$$\begin{aligned} & 1 \leq m + 1 \leq n \wedge \{k_1, \dots, k_{m+1}\} \subseteq \{1, \dots, n\} \wedge \\ & \quad |\{k_1, \dots, k_{m+1}\}| = m + 1 \wedge R_{k_1} \in S_\diamond \wedge \\ & \quad \forall i \in \{1, \dots, m+1\} : src(V'_{k_i}) = V_{k_i} \wedge \\ & \quad \forall i \in \{1, \dots, m\} : \big((R_{k_i}, R_{k_{i+1}}) \in \diamond \wedge \exists \langle V'_{k_i}, \dots, V'_{k_{m+1}} \rangle\big) \\ \Leftrightarrow\ & P_V(k_1, \dots, k_{m+1}) \end{aligned}$$

The proof is analogous for $P_H(k_1, \dots, k_m) \Rightarrow \big(R_{k_m} \in E_\square \vee \exists k_{m+1} : P_H(k_1, \dots, k_{m+1})\big)$.

(3) Assume that $P_V(k_1, \dots, k_m)$ holds and let $i \in \{1, \dots m\}$ be a corresponding index, then the following holds:

$$\begin{aligned} & src(V'_{k_i}) = V_{k_i} \\ \Rightarrow\ & V'_{k_i} \in V_D \wedge src(V'_{k_i}) = V_{k_i} \text{ is not a leaf of } VS^+ \end{aligned}$$

Distinguish the cases $R_{k_i} \in E_\diamond$ and $R_{k_i} \notin E_\diamond$. First assume that $R_{k_i} \in E_\diamond$ holds, then the following holds:

$$\begin{aligned}
&\exists U \in \big\{\{R_{k_i}\}\big\} : \forall u \in U : \exists v' \in V_D : src(v') = u \\
\Rightarrow\ &\exists v' \in V_D : src(v') = R_{k_i} \\
\Rightarrow\ &R_{k_i} \in src(V_D) \\
\Rightarrow\ &R_{k_i} \in q
\end{aligned}$$

Now assume that $R_{k_i} \notin E_\diamond$ holds, then the following holds:

$$\begin{aligned}
&\exists U \in \big\{\{V_j, R_{k_i}\} | (R_{k_i}, R_j) \in \diamond\big\} : \forall u \in U : \\
&\quad \exists v' \in V_D : src(v') = u \\
\Rightarrow\ &\exists v' \in V_D : src(v') = R_{k_i} \\
\Rightarrow\ &R_{k_i} \in src(V_D) \\
\Rightarrow\ &R_{k_i} \in q
\end{aligned}$$

In total, $P_V(k_1, \ldots, k_m) \Rightarrow \forall i \in \{1, \ldots, m\} : R_{k_i} \in q$ holds. The proof for the horizontal part $P_H(k_1, \ldots, k_m) \Rightarrow \forall i \in \{1, \ldots, m\} : R_{k_i} \in q$ is analogous. □

This concludes the overall proof for operation *read*. □

Proof for Operation Write. We show that the equivalence $Q_W = quorums^+_{write}(VS^+, w, \emptyset)$ for write quorums also holds, by showing two subset relations separately.

$\subseteq$:

Let $q \in Q_W$ be a write quorum of the grid structure GS. Then, q contains the nodes of a vertical path *and* a horizontal path and the following holds:

$$\begin{aligned}
&q = \{p_1, \ldots, p_m\} \wedge \forall i \in \{1, \ldots, m\} : p_i \in R \wedge \\
&\exists\ I = \{i_1, \ldots, i_r\},\ J = \{j_1, \ldots, j_s\} \subseteq \{1, \ldots, m\} : \big(I \cup J = \{1, \ldots m\} \wedge \\
&\quad \forall k \in \{1, \ldots, r-1\} : (p_{i_k}, p_{i_{k+1}}) \in \diamond \wedge p_{i_1} \in S_\diamond \wedge p_{i_r} \in E_\diamond \\
&\quad \forall k \in \{1, \ldots, s-1\} : (p_{j_k}, p_{j_{k+1}}) \in \square \wedge p_{j_1} \in S_\square \wedge p_{j_s} \in E_\square\big)
\end{aligned}$$

Additionally, we assume w.l.o.g.:

$$\begin{aligned}
\forall i \in I \setminus \{i_1\} : p_i \notin S_\diamond \wedge \forall i \in I \setminus \{i_r\} : p_i \notin E_\diamond \wedge \\
\forall j \in J \setminus \{j_1\} : p_i \notin S_\square \wedge \forall j \in J \setminus \{j_s\} : p_i \notin E_\square
\end{aligned}$$

Now consider the tree $D = (V_D, \lhd_D)$ with

$$\begin{aligned}
V_D =& \{G', V', v'_1, \ldots, v'_r, p'_1, \ldots, p'_r, H', h'_1, \ldots, h'_s, p''_1, \ldots, p''_s\} \\
\lhd_D =& \{(G', V'), (G', H'), (V', v'_1), (H', h'_1)\} \cup \\
&\{(v'_k, v'_{k+1}) | 1 \le k \le r-1\} \cup \{(v'_k, p'_k) | 1 \le k \le r\} \cup \\
&\{(h'_k, h'_{k+1}) | 1 \le k \le s-1\} \cup \{(h'_k, p''_k) | 1 \le k \le s\}
\end{aligned}$$

for which we show that it is a derivation tree, by means of the function *src* defined as follows:

$$\begin{aligned}
src(G') &:= G \\
src(V') &:= V^0 \\
src(v'_k) &:= V_{map(i_k)} \quad \forall k \in \{1, \dots, r\} \\
src(p'_k) &:= R_{map(i_k)} \quad \forall k \in \{1, \dots, r\} \\
src(H') &:= H^0 \\
src(h'_k) &:= H_{map(j_k)} \quad \forall k \in \{1, \dots, s\} \\
src(p''_k) &:= R_{map(j_k)} \quad \forall k \in \{1, \dots, s\}
\end{aligned}$$

$$\text{where } map : \{1, \dots, m\} \to \{1, \dots, n\}$$
$$map(k) = a :\Leftrightarrow p_k = R_a$$

Note, that *src* is not an injective function this time. There is at least one $R_k \in R$ which is the image of two different nodes of D. Now we show that D is a derivation tree of VS^+ for q in operation *write*:

(1) Let $v_1, v_2 \in V_D$ be nodes of D with $v_1 \lhd_D v_2$:

$$\begin{aligned}
G' \lhd_D V' :&\quad G \lhd V^0 \Rightarrow src(G') \lhd src(V') \\
G' \lhd_D H' :&\quad G \lhd H^0 \Rightarrow src(G') \lhd src(H') \\
V' \lhd_D v'_1 :&\quad p_{i_1} \in S_\diamond \Rightarrow R_{map(i_1)} \in S_\diamond \Rightarrow V^0 \lhd V_{map(i_1)} \Rightarrow src(V') \lhd src(v'_1) \\
H' \lhd_D h'_1 :&\quad p_{j_1} \in S_\square \Rightarrow R_{map(j_1)} \in S_\square \Rightarrow H^0 \lhd H_{map(j_1)} \Rightarrow src(H') \lhd src(h'_1) \\
\forall k \in \{1,\dots,r-1\}: & \\
v'_k \lhd_D v'_{k+1} :&\quad (p_{i_k}, p_{i_{k+1}}) \in \diamond \Rightarrow (R_{map(i_k)}, R_{map(i_{k+1})}) \in \diamond \\
&\quad \Rightarrow V_{map(i_k)} \lhd V_{map(i_{k+1})} \Rightarrow src(v'_k) \lhd src(v'_{k+1}) \\
\forall k \in \{1,\dots,r\}: & \\
v'_k \lhd_D p'_k :&\quad 1 \le k \le r \Rightarrow 1 \le i_k \le m \Rightarrow 1 \le map(i_k) \le n \\
&\quad \Rightarrow V_{map(i_k)} \lhd R_{map(i_k)} \Rightarrow src(v'_k) \lhd src(p'_k) \\
\forall k \in \{1,\dots,s-1\}: & \\
h'_k \lhd_D h'_{k+1} :&\quad (p_{j_k}, p_{j_{k+1}}) \in \square \Rightarrow (R_{map(j_k)}, R_{map(j_{k+1})}) \in \square \\
&\quad \Rightarrow H_{map(j_k)} \lhd H_{map(j_{k+1})} \Rightarrow src(h'_k) \lhd src(h'_{k+1}) \\
\forall k \in \{1,\dots,s\}: & \\
h'_k \lhd_D p''_k :&\quad 1 \le k \le s \Rightarrow 1 \le j_k \le m \Rightarrow 1 \le map(j_k) \le n \\
&\quad \Rightarrow H_{map(j_k)} \lhd R_{map(j_k)} \Rightarrow src(h'_k) \lhd src(p''_k)
\end{aligned}$$

(2) Let $v_1, v_2 \in V_D$ be nodes of D with $\exists \langle v_1, \dots, v_2 \rangle$. Assume that $\neg\big(src(v_1) \neq src(v_2)\big)$

holds:

$$
\begin{aligned}
&\exists\langle v_1,\dots,v_2\rangle \wedge \mathit{src}(v_1)=\mathit{src}(v_2)\\
\Rightarrow\ &\exists\langle v_1,\dots,v_2\rangle \wedge \big(v_1=v_2 \vee \exists k\in\{1,\dots,n\} : \mathit{src}(v_1)=R_k=\mathit{src}(v_2)\big)\\
\Rightarrow\ &\big(\exists\langle v_1,\dots,v_2\rangle \wedge v_1=v_2\big) \vee\\
&\big(\exists\langle v_1,\dots,v_2\rangle \wedge \exists k\in\{1,\dots,n\} : \mathit{src}(v_1)=R_k=\mathit{src}(v_2)\big)\\
\Rightarrow\ &\big(v_1\neq v_2 \wedge v_1=v_2\big) \vee \big(v_1 \text{ is not a leaf of } D \wedge\\
&\quad(\exists k\in\{1,\dots,r\} : v_1=p'_k \vee \exists k\in\{1,\dots,s\} : v_1=p''_k)\big)\\
\Rightarrow\ &\mathit{false} \vee \mathit{false} \quad \lightning
\end{aligned}
$$

Thus, in total, $\mathit{src}(v_1)\neq\mathit{src}(v_2)$ holds.

(3) G' is the root w_D of D, since G' is the only node without predecessor. Therefore, $\mathit{src}(w_D)=\mathit{src}(G')=G=w$ holds.

(4) For the write quorum $q=\{p_1,\dots,p_m\}$, the following holds:

$$
\begin{aligned}
&q=\{p_1,\dots,p_m\}=\{p_k|k\in I\cup J\}=\{p_i|i\in I\}\cup\{p_j|j\in J\}\\
=&\{R_{map(i_k)}|k\in\{1,\dots,r\}\}\cup\{R_{map(j_k)}|k\in\{1,\dots,s\}\}\\
=&\emptyset\cup\emptyset\cup\emptyset\cup\big((\{R_{map(i_k)}|k\in\{1,\dots,r\}\}\cup\{R_{map(j_k)}|k\in\{1,\dots,s\}\})\cap R\big)\\
=&(\{G,V^0,H^0\}\cap R)\cup(\{V_{map(i_k)}|k\in\{1,\dots,r\}\}\cap R)\cup\\
&\quad(\{H_{map(j_k)}|k\in\{1,\dots,s\}\}\cap R)\cup\\
&\quad\big((\{R_{map(i_k)}|k\in\{1,\dots,r\}\}\cup\{R_{map(j_k)}|k\in\{1,\dots,s\}\})\cap R\big)\\
=&\big(\{G,V^0,H^0\}\cup\{V_{map(i_k)}|k\in\{1,\dots,r\}\}\cup\{H_{map(j_k)}|k\in\{1,\dots,s\}\}\cup\\
&\quad\{R_{map(i_k)}|k\in\{1,\dots,r\}\}\cup\{R_{map(j_k)}|k\in\{1,\dots,s\}\}\big)\cap R\\
=&\mathit{src}(\{G',V',H',v'_1,\dots,v'_m,h'_1,\dots,h'_s,p'_1,\dots,p'_r,p''_1,\dots,p''_s\})\cap R\\
=&\mathit{src}(V_D)\cap R
\end{aligned}
$$

(5) Let $v\in V_D$ be a node of D and let $g=\mathit{idx}(\mathit{src}(v))$ denote the index of the source of

v in the sequence $\underline{v}$ of VS^+:

$$
\begin{aligned}
v = G' :\ & \{src(b) | G' \lhd_D b\} = \{src(V'), src(H')\} = \{V^0, H^0\} \text{ is } \textit{2}\text{-minimal} \wedge \\
& \underline{q_{write}}_g = 2 \\
v = V' :\ & \{src(b) | V' \lhd_D b\} = \{src(v'_1)\} = \{V_{map(i_1)}\} \text{ is } \textit{1}\text{-minimal} \wedge \underline{q_{write}}_g = 1 \\
\overset{k \in \{1,\ldots,r-1\}}{v = v'_k} :\ & \{src(b) | v'_k \lhd_D b\} = \{src(v'_{k+1}), src(p'_k))\} = \{V_{map(i_{k+1})}, R_{map(i_k)}\} \\
& \text{is } \textit{n+1}\text{-minimal} \wedge \underline{q_{write}}_g = n+1 \\
v = v'_r :\ & \{src(b) | v'_r \lhd_D b\} = \{src(p'_r)\} = \{R_{map(i_r)}\} \text{ is } \textit{n}\text{-minimal} \wedge \underline{q_{write}}_g = n \\
\overset{k \in \{1,\ldots,r\}}{v = p'_k} :\ & src(p'_k) = R_{map(i_k)} = p_{i_k} \in q \wedge src(p'_k) = R_{map(i_k)} \text{ is a leaf of } VS^+ \\
v = H' :\ & \{src(b) | H' \lhd_D b\} = \{src(h'_1)\} = \{H_{map(j_1)}\} \text{ is } \textit{1}\text{-minimal} \wedge \underline{q_{write}}_g = 1 \\
\overset{k \in \{1,\ldots,s-1\}}{v = h'_k} :\ & \{src(b) | h'_k \lhd_D b\} = \{src(h'_{k+1}), src(p''_k))\} = \{H_{map(j_{k+1})}, R_{map(j_k)}\} \\
& \text{is } \textit{n+1}\text{-minimal} \wedge \underline{q_{write}}_g = n+1 \\
v = h'_s :\ & \{src(b) | h'_s \lhd_D b\} = \{src(p''_s)\} = \{R_{map(j_s)}\} \text{ is } \textit{n}\text{-minimal} \wedge \underline{q_{write}}_g = n \\
\overset{k \in \{1,\ldots,s\}}{v = p''_k} :\ & src(p''_k) = R_{map(j_k)} = p_{j_k} \in q \wedge src(p''_k) = R_{map(j_k)} \text{ is a leaf of } VS^+
\end{aligned}
$$

In total, $\exists$ derivation tree D of VS^+ for q in *write* holds. Thus, due to Theorem 1, $q \in quorums^+_{write}(VS^+, w, \emptyset)$ holds. $\square$

$\supseteq$:

Let $q \in quorums^+_{write}(VS^+, w, \emptyset)$ be a write quorum of VS^+. Due to Theorem 1, $\exists$ derivation tree D of VS^+ for q in *write*. We prove the following three statements:

(1) $\exists k_1 \in \{1, \ldots, n\} : P_V(k_1) \wedge \exists k'_1 \in \{1, \ldots, n\} : P_H(k'_1)$

(2) $P_V(k_1, \ldots, k_m) \Rightarrow \big(R_{k_m} \in E_\diamond \vee \exists k_{m+1} : P_V(k_1, \ldots, k_{m+1})\big) \quad \wedge$
$P_H(k_1, \ldots, k_m) \Rightarrow \big(R_{k_m} \in E_\square \vee \exists k_{m+1} : P_H(k_1, \ldots, k_{m+1})\big)$

(3) $P_V(k_1, \ldots, k_m) \Rightarrow \forall i \in \{1, \ldots, m\} : R_{k_i} \in q \quad \wedge$
$P_H(k_1, \ldots, k_m) \Rightarrow \forall i \in \{1, \ldots, m\} : R_{k_i} \in q$

Under the assumption that (1), (2), and (3) hold, the following holds:

$$
\begin{aligned}
&\exists\, k_1, \ldots, k_m \in \{1, \ldots, n\} : \\
&\quad \big(P_V(k_1, \ldots, k_m) \wedge R_{k_m} \in E_\diamond \wedge \forall i \in \{1, \ldots, m\} : R_{k_i} \in q\big) \wedge \\
&\exists\, k'_1, \ldots, k'_l \in \{1, \ldots, n\} : \\
&\quad \big(P_H(k'_1, \ldots, k'_l) \wedge R_{k'_l} \in E_\square \wedge \forall i \in \{1, \ldots, l\} : R_{k'_i} \in q\big) \\
&\Rightarrow \big(R_{k_1} \in S_\diamond \wedge \forall i \in \{1, \ldots, m-1\} : (R_{k_i}, R_{k_{i+1}}) \in \diamond \wedge R_{k_m} \in E_\diamond \wedge \\
&\qquad \forall i \in \{1, \ldots, m\} : R_{k_i} \in q\big) \wedge \\
&\quad \big(R_{k'_1} \in S_\square \wedge \forall i \in \{1, \ldots, l-1\} : (R_{k'_i}, R'_{k_{i+1}}) \in \square \wedge R_{k'_m} \in E_\square \wedge \\
&\qquad \forall i \in \{1, \ldots, l\} : R_{k'_i} \in q\big) \\
&\Rightarrow q \text{ contains a vertical path in } GS \qquad \wedge \\
&\quad q \text{ contains a horizontal path in } GS \\
&\Rightarrow q \in Q_W
\end{aligned}
$$

Thus, it is left to prove the statements (1), (2), and (3):

(1)

$$
\begin{aligned}
&G' = w_D \in V_D \wedge \mathit{src}(w_D) = w = G \\
&\Rightarrow G' \in V_D \wedge \mathit{src}(G') = G \text{ is not a leaf of } VS^+ \\
&\Rightarrow \exists U \in \big\{\{V^0, H^0\}\big\} : \forall u \in U : \exists v' \in V_D : \mathit{src}(v') = u \\
&\Rightarrow \exists v' \in V_D : \mathit{src}(v') = V^0 \wedge \exists v' \in V_D : \mathit{src}(v') = H^0
\end{aligned}
$$

Let $V', H' \in V_D$ be such v', then the following holds:

$$
\begin{aligned}
&V' \in V_D \wedge \mathit{src}(V') = V^0 \text{ is not a leaf of } VS^+ \wedge \\
&\quad H' \in V_D \wedge \mathit{src}(H') = H^0 \text{ is not a leaf of } VS^+ \\
&\Rightarrow \exists U \in \big\{\{V_k\} | R_k \in S_\diamond\big\} : \forall u \in U : \exists v' \in V_D : \mathit{src}(v') = u \wedge \\
&\quad \exists U \in \big\{\{H_k\} | R_k \in S_\square\big\} : \forall u \in U : \exists v' \in V_D : \mathit{src}(v') = u \\
&\Rightarrow \exists k_v \in \{1, \ldots, n\} : (R_k \in S_\diamond \wedge \exists v'_v \in V_D : \mathit{src}(v'_v) = V_{k_v}) \wedge \\
&\quad \exists k_h \in \{1, \ldots, n\} : (R_k \in S_\square \wedge \exists v'_h \in V_D : \mathit{src}(v'_h) = H_{k_h})
\end{aligned}
$$

Let $k_1 \in \{1, \ldots, n\}$ be such a k_v and let V'_{k_1} be the corresponding v'_v, then the following holds:

$$
\begin{aligned}
&k_1 \in \{1, \ldots, n\} \wedge R_{k_1} \in S_\diamond \wedge \mathit{src}(V'_{k_1}) = V_{k_1} \\
&\Rightarrow 1 \leq 1 \leq n \wedge \{k_1\} \subseteq \{1, \ldots, n\} \wedge |\{k_1\}| = 1 \wedge R_{k_1} \in S_\diamond \wedge \\
&\quad \forall i \in \{1\} : \mathit{src}(V'_{k_i}) = V_{k_i} \wedge \\
&\quad \forall i \in \{1, \ldots, 1-1\} = \emptyset : \big((R_{k_i}, R_{k_{i+1}}) \in \diamond \wedge \exists \langle V'_{k_i}, \ldots, V'_{k_{i+1}} \rangle\big) \\
&\Rightarrow P_V(k_1)
\end{aligned}
$$

The arguments for $\exists k_h \in \{1, \ldots, n\} : (R_k \in S_\square \wedge \exists v'_h \in V_D : src(v'_h) = H_{k_h})$ with k'_1 being such a k_h implies $P_H(k'_1)$ are analogous. Overall $\exists k_1 \in \{1, \ldots, n\} : P_V(k_1) \wedge \exists k'_1 \in \{1, \ldots, n\} : P_H(k'_1)$ holds.

(2) Analog to *read*.

(3) Analog to *read*. □

This concludes the overall proof for operation *write*. □

7 Conclusion

The main goal of paper [11] was to model voting structures for any path-based replication strategy, in order to contribute to the *automated* design of *application-optimized* replication strategies for different scenarios. Therefore, we extended the concept of voting structures and proposed a design pattern for arbitrary path-based replication strategies in the extended concept [11]. This opens up many new opportunities for improving the mechanism, presented in [10], which up to now only uses classic voting structures for the automatic design of scenario-based application-optimized replication strategies through genetic programming. With the proofs, presented in this technical report, the correctness of the concept of *extended* voting structures is ensured. This constitutes a reliable conceptional base on which we can justify further research on the development of a new approach for the automatic design of scenario-based application-optimized replication strategies through genetic programming by using *extended* voting structures.

Acknowledgements

This research is funded by the German Research Foundation through the Research Training Group DFG-GRK 1765/2 "SCARE – System Correctness under Adverse Conditions", https://www.uni-oldenburg.de/en/scare/

References

[1] P. A. Bernstein, V. Hadzilacos, and N. Goodman, *Concurrency Control and Recovery in Database Systems.* Addison-Wesley, 1987.

[2] R. H. Thomas, "A Majority Consensus Approach to Concurrency Control for Multiple Copy Databases," *ACM Trans. Database Syst.*, vol. 4, no. 2, pp. 180–209, June 1979.

[3] P. A. Bernstein and N. Goodman, "An Algorithm for Concurrency Control and Recovery in Replicated Distributed Databases," *ACM Trans. Database Syst.*, vol. 9, no. 4, pp. 596–615, Dec. 1984.

[4] H. Garcia-Molina and D. Barbara, "How to Assign Votes in a Distributed System," *J. ACM*, vol. 32, no. 4, pp. 841–860, Oct. 1985. [Online]. Available: https://doi.org/10.1145/4221.4223

[5] S. Y. Cheung, M. H. Ammar, and M. Ahamad, "The grid protocol: a high performance scheme for maintaining replicated data," in *Proc. of the Sixth International Conference on Data Engineering*, Feb 1990, pp. 438–445.

[6] C. Wu and G. G. Belford, "The triangular lattice protocol: a highly fault tolerant and highly efficient protocol for replicated data," in *Proc. of the 11th Symposium on Reliable Distributed Systems*, 1992, pp. 66–73.

[7] O. Theel and H. Pagnia, "Optimal Replica Control Protocols Exhibit Symmetric Operation Availabilities," in *Proc. of the 28th International Symposium on Fault-Tolerant Computing (FTCS-28)*. IEEE, June 1998, pp. 252–261.

[8] M. Arai, T. Suzuki, M. Ohara, S. Fukumoto, K. Iwasaki, and Hee Yong Youn, "Analysis of read and write availability for generalized hybrid data replication protocol," in *Proc. of the 10th IEEE Pacific Rim International Symposium on Dependable Computing*, 2004, pp. 143–150.

[9] S. M. A. Bokhari and O. Theel, "A Flexible Hybrid Approach to Data Replication in Distributed Systems," in *Intelligent Computing*, K. Arai, S. Kapoor, and R. Bhatia, Eds. Springer International Publishing, 2020, pp. 196–207.

[10] ——, "Design of Scenario-based Application-optimized Data Replication Strategies through Genetic Programming," in *Proc. of the 12th International Conference on Agents and Artificial Intelligence (ICAART 2020)*, vol. 2, INSTICC. SciTePress, 2020, pp. 120–129.

[11] C. Linder and O. Theel, "Extending the Concept of Voting Structures to Support Path-Based Replication Strategies," in *Proc. of the IEEE 26th Pacific Rim International Symposium on Dependable Computing (PRDC'21)*, 2021 (To be Published).

[12] O. Theel, "General Structured Voting: A Flexible Framework for Modelling Cooperations," in *Proc. of the 13th International Conference on Distributed Computing Systems*. IEEE, May 1993, pp. 227–236.

[13] A. Seifert, "Entwurf einer Simulationsumgebung für Replikationsstrategien in verteilten Systemen (in German)," Student paper, Darmstadt University of Technology, Dept. of Computer Science, Operating Systems Group, Germany, 1996.

[14] O. Theel and H. Pagnia-Koch, "General Design of Grid-based Data Replication Schemes Using Graphs and a Few Rules," in *Proc. of the 15th International Conference on Distributed Computing Systems*. IEEE, May 1995, pp. 395–403.

[15] C. Storm, *Specification and Analytical Evaluation of Heterogeneous Dynamic Quorum-Based Data Replication Schemes*. Springer Science & Business Media, 2012.

[16] C. Linder, "Extension of the Concept of Voting Structures to Support Path-Based Replication Strategies," Master's thesis, Carl von Ossietzky University of Oldenburg, Dept. of Computer Science, System Software and Distributed Systems Group, Germany, 2020.