On Extend-Only Directed Posets and Derived Byzantine-Tolerant Replicated Data Types

Florian Jacob
florian.jacob@kit.edu
Karlsruhe Institute of Technology
Karlsruhe, Germany

Hannes Hartenstein
hannes.hartenstein@kit.edu
Karlsruhe Institute of Technology
Karlsruhe, Germany

Abstract

We uncover the extend-only directed posets (EDP) structure as unification of recently discussed DAG-based Byzantine-tolerant conflict-free replicated data types (CRDT). We also derive an EDP-based key-value map, and give an outlook on an EDP-based systemic access control CRDT as formalization of the CRDT used in the Matrix messaging system.

CCS Concepts: • **Security and privacy → Distributed systems security**; *Access control*; • **Software and its engineering → Consistency**.

Keywords: Conflict-Free Replicated Data Types, Strong Eventual Consistency, Byzantine Fault Model, Matrix Event Graph

ACM Reference Format:
Florian Jacob and Hannes Hartenstein. 2023. On Extend-Only Directed Posets and Derived Byzantine-Tolerant Replicated Data Types. In *10th Workshop on Principles and Practice of Consistency for Distributed Data (PaPoC '23), May 8, 2023, Rome, Italy*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3578358.3591333

1 Introduction

Recently, we noticed [7] that the conflict-free replicated data type (CRDT) of the Matrix decentralized messaging system [4] retains its CRDT property in environments with Byzantine nodes, i.e., nodes that arbitrarily deviate from the expected protocol. Kleppmann [8] showed that arbitrary crash-fault tolerant CRDTs can be transformed to tolerate arbitrary numbers of Byzantine nodes. These proposals have in common that events are managed in a shared graph object: events are independently generated (available under partition), but chained to previous events known to the replica.

The shared objects essentially show the following dynamic behavior. All replicas start with the same genesis event as initial state. Replicas append new events to all events without learned descendants, i.e., the "most recent" events, without coordinating with other replicas. Replicas then exchange updates, to reach a consistent state again. When replicas append events concurrently, branches occur, which leads to a tree-like structure. When replicas learn of branches, they will join them on their next event. Events consist of a payload and hashes of previous events, which ensure integrity and strong eventual consistency against Byzantine equivocation [5, 8]. An example evolution of such an object is illustrated in Fig. 1.

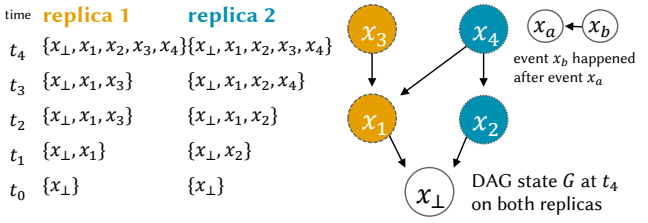

Figure 1. Example graph state G at a point in time t_4 as seen by both replicas. First, the replicas send events x_1 and x_2 concurrently. When replica 2 sends x_4, it has learned x_1 and therefore puts $\{x_1, x_2\}$ as happened-after parents. Replica 1 has not learned x_2 yet, and thereby appends x_3 only to x_1.

While in previous work, the corresponding objects were described as directed acyclic graphs (DAGs), in this work-in-progress paper we propose a more fundamental, unifying CRDT formalization via *directed partially-ordered sets* (directed posets), called Extend-only Directed Posets (EDPs), that shows the following advantages:

a) The set-theoretic formalization captures the essence of Byzantine-tolerant DAG CRDTs, revealing a state-based EDP that does not require crypto elements, and a hash-based operation-based construction as efficiency optimization. In contrast to DAG-based variants, EDPs use standard mathematical notions and fit the theory of CRDTs that is largely formulated in terms of set theory [12]. Due to the close relation of set and lattice theory to boolean algebra [11], we believe EDPs facilitate formal verification in future work.

b) Based on the generic EDP, other types can be derived and/or constructed by composition that can then more easily be shown to have the CRDT property in Byzantine environment. We present a derived map type, and an outlook towards a composed CRDT for systemic access control as formalization of the CRDT used in Matrix.

PaPoC '23, May 8, 2023, Rome, Italy

ACM ISBN 979-8-4007-0086-6/23/05…$15.00
https://doi.org/10.1145/3578358.3591333

Related work made use of sets instead of DAGs in Byzantine environments, however, did not treat and generalize them as CRDTs [3]. In the following sections, we will define EDPs and indicate why they represent a solid basis for the class of Byzantine-tolerant CRDTs discussed above.

2 Extend-only Directed Posets (EDPs)

2.1 Basics and Notations

The EDP CRDT is based on relational structures, for which we now introduce mathematical basics and notation.

A relational structure $S = (X, R)$ is a tuple of a ground set X and a relation $R \subseteq X \times X$. To facilitate notation, we sometimes write $S.X$ and $S.R$ to denote set X and relation R of $S = (X, R)$. A partially-ordered set (poset) is a relational structure that is reflexive ($\forall a \in X\colon (a, a) \in R$), transitive ($\forall a, b, c \in X\colon (c, b) \in R \land (b, a) \in R \Rightarrow (c, a) \in R$), and antisymmetric ($\forall a, b \in X\colon (a, b) \in R \land (b, a) \in R) \Rightarrow a = b$). While a $\leq$-like R is usual in mathematics, we use a $\geq$-like R in these definitions for consistency with the rest of the paper. If S is a poset, then R is called a partial order relation, or partial ordering. If S is also strongly connected ($\forall a, b \in X\colon (a, b) \in R \lor (b, a) \in R$), S is called a linearly-ordered set.

A reflexive and transitive relational structure S is called a downward-directed set if for any two elements, the set contains a lower bound, i.e., $\forall a, b \in X\colon \exists x_{lb} \in X$ s.t. $(a, x_{lb}) \in R \land (b, x_{lb}) \in R$. A finite, downward-directed poset is directed towards its unique least element $x_\perp$, also denoted as $S^\perp$, i.e., $x_\perp \in X$ and $\forall x \in X\colon (x, x_\perp) \in R$. Conversely, a finite upward-directed poset is directed towards its top element $S^\top = x_\top \in X$, i.e., $\forall x \in X\colon (x_\top, x) \in R$. A both downward- and upward-directed finite poset is said to be a poset bounded by $x_\perp$ and $x_\top$. The set of maximal elements of S is $\max(S) = \{m \in X | \forall x \in X\colon (x, m) \in R \Rightarrow (m, x) \in R\}$. Conversely, the set of minimal elements of S is $\min(S) = \{m \in X | \forall x \in X\colon (m, x) \in R \Rightarrow (x, m) \in R\}$.

A relational structure $S' = (X', R')$ is called an extension of another relational structure S if $X \subseteq X'$ and $R = R'|_X$, where the restriction $R|_A$ is defined as usual as $R|_A = R \cap (A \times A)$. We call S' an upward extension of a downward-directed poset S if additionally $S'^\perp = S^\perp$. The downward closure $y^{\downarrow S}$ of an element $y \in S.X$ is defined as $y^{\downarrow S} = \{c \in X | (y, c) \in R\}$. The downward closure $Y^{\downarrow S}$ of a subset $Y \subseteq X$ is generalized from the single-element downward closure as $Y^{\downarrow S} = \bigcup_{y \in Y} y^{\downarrow S}$. The upward closure of elements and subsets of X is defined correspondingly, $y^{\uparrow S} = \{c \in X | (c, y) \in R\}$. The set of maximal lower bounds of an element $y \in X$ is the set of maximal elements of the downward closure of y without y itself, $\text{mlb}(y) = \max(y^{\downarrow S} \setminus \{y\})$.

2.2 Specification

To build an append-only CRDT (as sketched in the introduction) that tolerates an arbitrary number of Byzantine nodes, the key idea is to replace an event x with its downward closure, i.e., adding all the relations to previous events. Thereby, a byzantine node cannot create inconsistent relations anymore — it can only create spam additions. The EDP RDT will be defined, therefore, as 'higher-order' directed posets, i.e., will represent directed posets of directed posets. The resulting EDP RDT can then easily be shown to represent a state-based CRDT, for which an operation-based version can be constructed by using cryptographic hash functions as order-guaranteeing mechanism. For clarity, we denote as Basic Directed Posets (BDP) the directed posets that are used as 'base layer' to construct the EDP RDT.

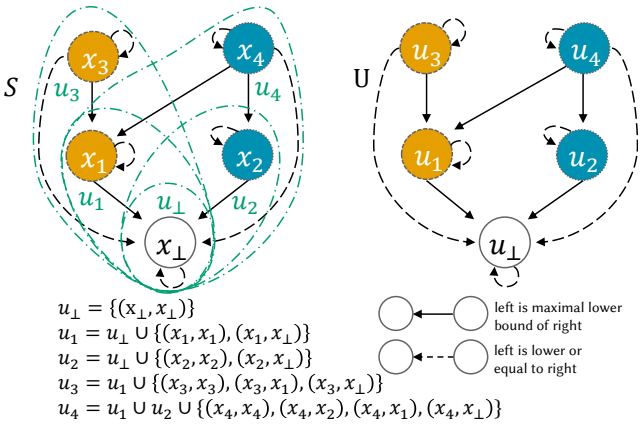

Figure 2. A BDP state S and equivalent EDP state U based on Fig. 1. An upward extension $u_i \in U$ is the set of its enclosed relations in S, to represent the formation history of $x_i \in S$.

To fix a BDP, one selects a universe $\mathbb{X}$ of valid elements and a universe $\mathbb{R} \subseteq \mathcal{P}(\mathbb{X}^2)$ of valid relations. BDP states are finite, downward-directed posets $S = (X, R)$, $X \subseteq \mathbb{X}$, $R \in \mathbb{R}$, with a common bottom element $S^\perp$ (to be interpreted as the initial state of all correct replicas). As example, one can think of the universe $\mathbb{X}$ as application-layer messages, and the universe $\mathbb{R} \subseteq \mathcal{P}(\mathbb{X}^2)$ as causal orders, and a specific bootstrapping application-layer message as bottom element.

To define extend-only operations, we only want to allow single-element upward extensions of a BDP state S. We want finality as required validity criterion for extensions: Adding a new element to the BDP and adding the relations of the new element needs to be a single, atomic upward extension, i.e., cannot be changed later. The required finality property can be expressed as follows: when element x is the single element that has been added to gain state S, for all upward extensions S' of S needs to hold[1]: $x^{\downarrow S'} = x^{\downarrow S}$ and $R'|_{x^{\downarrow S'}} = R|_{x^{\downarrow S}}$.

To provide this finality notion, the obvious (but inefficient) approach is to bind a single-element upward extension, which extends $S = (X, R)$ with an element $x \in \mathbb{X}$, to

[1] In contrast, the upward closure of an element and their relations might be never final: $x^{\uparrow S'} \supseteq x^{\uparrow S}$ and $R'|_{x^{\uparrow S'}} \supseteq R|_{x^{\uparrow S}}$. A new upper element can always be in transit or purported to have been in transit.

all single-element upward extensions for elements y with $x \geq y$. This way, the history of extend-only operations is fixed and serves as identity-forming information for the new single-element upward extension. To express sets of 'formation histories' of BDP elements, one has to move to sets of posets. Thus, for the EDP definition, we move 'one level up' and map those BDP posets to elements of the EDP as done in the following two steps. An illustration of the BDP to EDP state mapping based on the state in Fig. 1 is found in Fig. 2.

Step 1. Let the directed poset $S' = (X', R')$ be the upward extension with a single element $x \in \mathbb{X}$ of a directed poset $S = (X, R)$ of the BDP. The downward closure $x^{\downarrow S'}$ with R' restricted to $x^{\downarrow S'}$ is a sub-poset $(x^{\downarrow S'}, R'|_{x^{\downarrow S'}}) \subseteq S'$ bounded by $x_\perp$ and x. Let $u_x := R'|_{x^{\downarrow S'}}$ denote this relation of such a BDP upward extension with x, and $X(u_x) := \{y \in \mathbb{X} | (y, y) \in u_x\} = x^{\downarrow S'}$ the set of reflexive pairs in u_x. We can derive from u_x the upward extension S' of S with x as $S' = (X \cup X(u_x), R \cup u_x)$. Thereby, as shorthand notation, we call u_x an upward extension as well. An upward extension $u_x \in \mathbb{R}$ is valid if $(X(u_x), u_x)$ forms a BDP sub-poset of S' bounded by $x_\perp$ and x. While we sometimes focus on the relation part of such sub-posets only, a formulation based on corresponding bounded sub-posets can be derived easily.

Step 2. We now move 'one level up' and define the EDP by using upward extensions u_x as elements and subset relations to form posets of those upward extensions. An EDP state $U \in \mathcal{P}(\mathbb{R})$ is the set of single-element upward extensions $U = \{u_\perp, u_1, \ldots\}$, i.e., the formation history of BDP state S. The initial EDP state is $U = \{u_\perp = \{(x_\perp, x_\perp)\}\}$, which is the upward extension of the empty set with the bottom element. An EDP state $U \in \mathcal{P}(\mathbb{R})$ is valid if $(U, \supseteq |_U)$ is a $u_\perp$-directed poset and $\forall u \in U\colon\ \mathrm{mlb}(u) \subsetneq U \wedge |X(u)| = |\{X(\bigcup \mathrm{mlb}(u))| + 1$, i.e., every upward extension in U also has its maximal lower bounds in U, and extends the BDP state with exactly one element that is not present in any of its maximal lower bounds. Fig. 3 provides an illustration of applying an upward extension, derived from the graph of Fig. 1. An EDP state U can be transformed back to a directed poset $S(U)$ on the underlying BDP via $S(U) = (X(\bigcup U), \bigcup U)$. The above EDP definition leads to a *state-based CRDT* with state space $\mathcal{P}(\mathbb{R})$ and set union $U_1 \cup U_2$ as join:

Theorem 1. *Assuming a connected component of all correct replicas and eventual communication within the component, the state-based EDP is a Conflict-free Replicated Data Type even in face of an arbitrary number of Byzantine replicas.*

Proof Sketch. For the state space $\mathcal{P}(\mathbb{R})$, set union $U_1 \cup U_2$ is the least upper bound of $U_1, U_2 \in \mathcal{P}(\mathbb{R})$, whereby $\mathcal{P}(\mathbb{R})$ and set union form a join-semilattice. Via periodic state gossiping, eventual delivery is fulfilled. Together with termination from the purely mathematical state join, the state-based EDP is a CRDT [14]. As updates only consist of a semilattice element, there is no metadata to forge, and Byzantine replicas are restricted in their actions: Invalid Byzantine updates are rejected as not part of the semilattice, and due to the full formation history protecting the integrity of valid updates, any Byzantine attempt to alter history, equivocate, or otherwise harm consistency, simply results in additional valid updates which were just not successfully sent to all replicas [5]. □

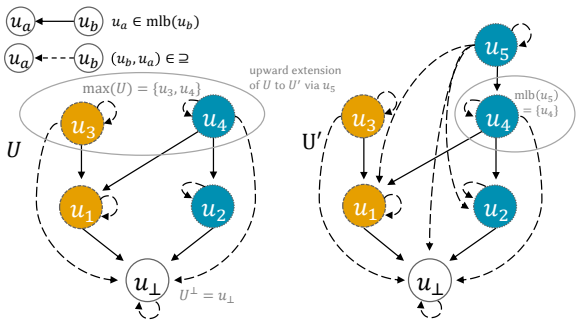

Figure 3. Example of a replica state U of an EDP, and state U' that results from an upward extension of U with u_5.

While modeling as directed posets might introduce some formalism, it uncovers the simple set-theoretic fundamentals, i.e., the state-based structure, of Byzantine-tolerant DAG CRDTs. Based on this set-theoretic view, Theorem 1 is easily shown without needing cryptographic building blocks like hash functions. However, from a practical point of view, a state-based EDP is highly inefficient: replicas continuously gossip their full state U, which will only increase in size. To reach the efficiency of DAGs with EDPs, we work up our way from the state-based formulation to an efficient operation-based EDP CRDT formulation that makes use of cryptographic hashes for integrity protection. The construction follows [7, 8], its purpose here is to clarify the relationship between operation-based and state-based formalizations. We need two optimizations for which we give an intuition now, and a formalization next. As first optimization, an upward extension u_y represents a sub-poset of S' bounded by $x_\perp$ and y, thus, sending only the relations of that sub-poset instead of the state U' is sufficient. For the *operation-based formulation*, we then compress the subset, reducing worst-case update size from 'depth' to 'width' of U': We define an operation o consisting of y and the set of hashed maximal lower bounds of u_y in U'. As $\mathrm{mlb}(u_y) \subseteq U$, u_y can be reconstructed based on operation o and existing state U.

To chain y to its maximal lower bounds, a preimage and collision resistant hash function $h(a, b)$ is used that returns the hash of its chained arguments. We recursively define the set of hashes of a set of maximal lower bounds U'_{mlb} as $H(U'_{\mathrm{mlb}}) := \{h(\max(X(u)), H(\mathrm{mlb}(u)))|u \in U'_{\mathrm{mlb}}\}$. To define the hash of the set of hashes in the second argument of h, one can concatenate its linearization gained by lexicographical sorting. An operation $o := (y, H_{\mathrm{mlb}})$ is then defined as compression of upward extension u via $o(u) := (\max(X(u)), H(\mathrm{mlb}(u)))$. The hashed maximal lower bound

set recursively protects the integrity of the directed subposet down to the bottom element $u_\perp$ against Byzantine nodes. To reconstruct upward extension u from operation o based on knowledge of current state U, we need the set of relations of all of $o.y$'s predecessors $P := (\bigcup U'_{\text{mlb}} \subseteq U | H(U'_{\text{mlb}}) = o.H_{mlb})$. Then, the upward extension is the union of the reflexive relation of y, the relation of y to its predecessor elements $X(P)$, and its transitive relation, i.e., its predecessor relations P. We formalize this reconstruction as $u(y, P) := \{y\}^2 \cup \{y\} \times X(P) \cup P$. An operation o can be locally applied as soon as all maximal lower bounds are part of the replica state, i.e., $o.H_{\text{mlb}} \subseteq H(U)$. An operation o is applied to state U by $U' = U \cup u(o.y, P)$, inheriting commutativity from set union, which makes for an operation-based CRDT with state space $\mathcal{P}(\mathbb{R})$. Update size is now bound to the maximum size of a maximal lower bound set of any element in the downward closure, instead of overall state size.

The operation-based EDP is given in Algorithm 1. Encountering an unsatisfied assertion, the algorithm stops processing the function and returns an error. Encountering an unsatisfied await, the algorithm interrupts to await its potential future satisfaction, without blocking subsequent function calls. The `extend` function is used to extend the current state U with a new upward extension $u \in \mathbb{R}$. The side-effect-free `generate` function generates update operation o and broadcast it to all replicas, including itself. Received broadcasts are processed by the `effect` function, which awaits all maximal lower bounds to be part of the current state before applying the operation. Byzantine tolerance is sketched in Section 2.4, and requires a weak resilient broadcast outlined next.

2.3 Resilient Broadcast of Operation-Based Updates

The operation-based EDP formulation does not need the strong guarantees of a crash-/Byzantine-fault reliable broadcast. As explained in [8], mere eventual delivery of updates is sufficient for such Byzantine Sybil-resistant CRDTs, as long as correct replicas form a connected component. Due to the shared bottom element and relation directedness, broadcasting the set of maximal elements is sufficient, as missing elements can be iteratively queried from other replicas and integrity-verified via the hash chain. An optimized broadcasting approach in this spirit is found in [8].

When the set of maximal upward extensions $\hat{U} = \max(U)$ of the replicas' current state U (or their space-efficient operation equivalents) is gossiped regularly, the broadcast is a state-based CRDT itself: The state space forms a join-semilattice with join of $\hat{U}_1$ and $\hat{U}_2$ being $\max(\hat{U}_1 \cup \hat{U}_2)$. While update size is close to EDP size in worst case, if all replicas are correct, update size converges quickly to approximately the number of involved replicas [7]. Byzantine replicas can send a large set of made-up extensions, but as the resulting state-based broadcast is a Byzantine Sybil-resistant CRDT [5], they can only attack performance but not correctness.

Algorithm 1 Op-Based EDP CRDT for BDP ($X \subseteq \mathbb{X}, R \in \mathbb{R}$)

Require: universe of valid BDP elements $\mathbb{X}$
Require: universe of valid BDP relations $\mathbb{R} \subseteq \mathcal{P}(\mathbb{X}^2)$
  state set of upward extensions $U \in \mathcal{P}(\mathbb{R})$
  initial $U \leftarrow \{u_\perp = \{(x_\perp, x_\perp)\}\}$
  query *bot* () : $u_\perp = U^\perp \in U$
  query *max* () : $U_{\max} = \max(U) \subseteq U$
  query *mlb* ($u \in U$) : $U_{\text{mlb}} = \text{mlb}(u) \subsetneq U$
  query S () : directed poset $S = (X \subseteq \mathbb{X}, R \in \mathbb{R})$
    $S \leftarrow (X(\bigcup U\}, \bigcup U)$

  update *extend* ($u \in \mathbb{R}$)
    generate ($u \in \mathbb{R}$)
      assert $\emptyset \neq u \notin U$
      assert $\emptyset \neq \text{mlb}(u) \subsetneq U$
      $o \leftarrow (\max(X(u)), H(\text{mlb}(u)))$
    effect ($o = (y \in \mathbb{X}, H_{\text{mlb}})$)
      await $H_{\text{mlb}} \subseteq H(U)$ ▷ await effect of updates $\subsetneq u$
      assert $H_{\text{mlb}} \neq \emptyset$
      $P \leftarrow \bigcup(U'_{\text{mlb}} \subseteq U | H(U'_{\text{mlb}}) = H_{mlb})$
      $U \leftarrow U \cup (\{y\}^2 \cup \{y\} \times X(P) \cup P)$

2.4 Op-based Byzantine Strong Eventual Consistency

To show Byzantine strong eventual consistency of the op-based EDP RDT, we use the strong eventual consistency (SEC) notion as defined in [9] consisting of three properties: a) **Self-update**, i.e., iff a correct replica generates an update, it applies that update to its own state; b) **Eventual update**, i.e., for any update applied by a correct replica, all correct replicas will eventually apply that update; and c) **Strong Convergence**, i.e., any two correct replicas that have applied the same set of updates are in the same state.

Lemma 1. *Let $U \in \mathcal{P}(\mathbb{R})$ be a directed poset corresponding to $S = (X \subseteq \mathbb{X}, R \in \mathbb{R})$, and U' (S' resp.) the resulting state after applying the update $o = (y, H_{\text{mlb}})$ for upward extension u. Then U' (S' resp.) are a partially-ordered extensions of U (S resp.) directed towards the same element $U^\perp$ ($S^\perp$ resp.).*

Proof Sketch. Invalid upward extensions are discarded via assertions and the hash function's preimage resistance, and then $U' = U$. Valid upward extensions only add a single element y to S. Due to P being the union of $u_\perp$-directed subposets and $\{y\} \times X(P) \subsetneq u$, $U'^\perp = U^\perp$ and also $S'^\perp = S^\perp$, i.e., $\perp$-directedness is kept. Due to the same argument, the partial-order properties also still hold for both U' and S'. □

Theorem 2. *Assuming a connected component of all correct replicas and eventual communication within the component, the operation-based EDP is a Conflict-free Replicated Data Type even in face of an arbitrary number of Byzantine replicas.*

Proof Sketch. **Self-update:** With a broadcast as described in Section 2.3, the broadcasting replica receives the update

without waiting for any acknowledgment. The `generate` function creates an update for which the await precondition in `effect` is immediately satisfied, which means that the `effect` function directly updates the replica's own state. Byzantine nodes have no attack vector to interfere with this process. **Eventual update:** Using a broadcast as described in Section 2.3, as soon as one correct replica receives an update, eventually every correct replica will receive the update. Due to eventual communication within the connected component of correct replicas, Byzantine nodes have no attack vector to interfere with eventual delivery of correct updates among correct replicas. Through the hash-chained maximal lower bounds, replicas can verify the integrity and completeness of the directed sub-poset of the downward closure up until the bottom element on every new update, without a Byzantine node being able to interfere. As soon as one correct replica can satisfy the await precondition for applying an update, i.e., has received the necessary downward closure for the element to add, it will eventually share the downward closure with all correct replicas, for which the delivery precondition is then also fulfilled eventually so that they can proceed with the effect function as well. **Strong convergence:** Due to Lemma 1 and commutativity of set union, all valid updates commute. Due to the `effect` function's assertions, only valid updates are applied. Due to an operation consisting of both its payload as well as the hashes of its maximal lower bounds, the integrity of the directed poset of the downward closure of an element is integrity-protected through hash chaining. Thereby, a Byzantine node that tries to attack consistency via equivocation with two operations (y_a, H^a_{mlb}) and (y_b, H^b_{mlb}) where either $y_a = y_b$ or $H^a_{\text{mlb}} = H^b_{\text{mlb}}$ is not successful, as all correct replicas will reject neither update as already received, but treat them as separate updates. As validity checks are deterministic and the same on all replicas, and the hash function is collision resistant, Byzantine replicas cannot get an update applied on only a proper subset of correct replicas. □

With eventual communication among correct replicas, neither the operation-based nor the state-based EDP require knowledge of the sending replica of updates to provide SEC. As their update functions do not depend on the identity of replicas that created an update, a forged but valid update is just *another* valid update. However, digital signatures are required as soon as the application requires set elements to contain an identifier for the sending replica, e.g., for access control, which we will discuss in Section 3.2.

2.5 Causal Extend-Only Directed Poset (CEDPs)

We denote the subtype of Extend-only Directed Posets with temporal events as set items together with their causal relation, in form of their happened-after-equal relation, as Causal Extend-only Directed Posets (CEDPs). As an EDP subtype, Theorem 2 also applies. Hash chaining can represent any upward-extend-only partial ordering, but inherently proves that the image happened-after the preimage, and thereby the happened-after relation in a Byzantine-tolerant way.

CEDPs are a set-based version, i.e., the reflexive-transitive closure, of causal Event DAG approaches like the Matrix Event Graph (MEG) [4]. A specific proof that the MEG is a Byzantine-tolerant CRDT is found in [7]. In [9], an Event DAG approach is used as transport layer to make arbitrary CRDTs Byzantine-tolerant. Their arguments for Byzantine tolerance and broadcast requirements also apply to CEDPs.

While the causal set approach to discrete, logical time is well-known in quantum physics [1], in distributed systems, the happened-before relation as defined by Lamport [10] is common. The happened-after-equal relation is the converse of the reflexive closure of the happened-before relation, and thereby the relations are interchangeable. The happened-after-equal relation however represents the direction of hash chaining and an ordering-based notion of equality. In Physics, an event is defined as a point in space-time. An operation $o = (x, H_{\text{mlb}})$ can be read as "event x happened at discrete logical time coordinate H_{mlb} and discrete space coordinate of the (implicit) replica identifier". The happened-after order is a superset of the causal order, i.e., the order in which events causally depend on each other. Either the application provides new events and their maximal lower bounds in causal order, or we use the happened-after order via $\max(U)$ as maximal lower bounds, which then covers causal order.

3 Replicated Data Types Derived from EDPs

3.1 EDP-based Maps (EPMs)

3.1.1 EPM Specification. Based on the EDP replicated data type, we derive a map replicated data type we call EPM. Essentially, the universe $\mathbb{X}$ now represents key-value pairs, and as before, the relation $\mathbb{R}$ defines how update operations are ordered. The EPM has a 'largest-element-wins' semantics with respect to an ordering of update operations based on $\mathbb{R}$.

Formally, we define $\mathbb{M} \subseteq \mathbb{X}$ as the set of valid key-value pairs of the form $(k \mapsto v)$, and a map $M \subseteq \mathbb{M}$ as 'injective' subset of all valid key-value pairs, i.e., a given key only has one unique associated value. We define the square bracket operator to query keys for map M as $(k \mapsto v) \in M \Leftrightarrow M[k] = v$ and a map update operator $\uplus$ for single-element upward extensions u that keeps injectiveness of M:

$$\forall u \in \mathbb{R}, x = \max(X(u)) :$$

$$M \uplus u := \begin{cases} M \setminus \{k \mapsto \_\} \cup \{k \mapsto v\} & \text{if } x = (k \mapsto v) \in \mathbb{M} \\ M & \text{if } x \in \mathbb{X} \setminus \mathbb{M} \end{cases}$$

The functions of the EPM replicated data type are given in Algorithm 2. The core of the algorithm is the *linearize* function, which linearizes a set $T \subseteq U$ partially-ordered by $\subseteq$ to a sequence T_n. We define a relation $R_{\|}$ of all pairs of upward extensions $(u_1, u_2) \in T^2$ that are 'parallel', i.e., cannot be compared using $\subseteq$, and are also minimal in the

sense that no elements smaller than u_1 exist that cannot be compared to u_2, and vice-versa for u_2. Using a preimage and collision resistant cryptographic hash function h, we define a strict linear order relation $R_h = \{(u_1, u_2) \in \mathbb{R}^2 | h(u_1) < h(u_2)\}$. The relation $R_\parallel \cap R_h$ then contains the necessary tie-breakings for the partial ordering $\subseteq$ to gain the linear ordering $R_l = (\subseteq \cup (R_\parallel \cap R_h))^+$. Hashes allow to resolve ties without Byzantine nodes being able to compromise results by choosing order (preimages) or breaking linearity (collisions).

Note that Matrix also provides maps based on the CEDP similar to Section 3.1 to assign additional attributes to replicas and replicated objects, and uses them to provide access control, which we will discuss in Section 3.2.

3.1.2 Byzantine Eventual Consistency Verification.

Theorem 3. *Assuming a connected component of all correct replicas and eventual communication within the component, an EPM is an operation-based Conflict-free Replicated Data Type even in face of an arbitrary number of Byzantine replicas.*

Proof Sketch. By reduction to the Byzantine strong eventual consistency of the underlying operation-based EDP. In a correct replica, the key-value pair is put in context through the set of maximal elements $\max(U)$ as maximal lower bounds of $(k \mapsto v)$, which satisfies the await-precondition and keeps the **Self-Update property**. Nothing has changed in the broadcasting and application of updates, and the `linearize` function does not interact with other replicas but takes all updates applied in the EDP into account, which keeps **Eventual Delivery**. The `get` and `linearize` functions as well as the map update operator $\uplus$ are deterministic and ignore invalid updates, so given the same downward-directed poset $(U, \supseteq)$, i.e. the same state of the EDP, they return the same map M, maintaining **Strong convergence**. □

Algorithm 2 Operation-Based EPM Replicated Data Type

state set of upward extensions $U \in \mathcal{P}(\mathbb{R})$
initial $U \leftarrow \{u_\perp = \{(x_\perp, x_\perp)\}\}$

update *put* $((k \mapsto v) \in \mathbb{M})$
  $y \leftarrow (k \mapsto v)$
  extend$(\ u(y, \bigcup \max(U))\ )$ ▷ function of Algorithm 1
function *linearize* $(T \subseteq U) : T_n \in \mathbb{R}^n$
  $R_\parallel \leftarrow \{(u_1, u_2) \in T^2 | u_1 \not\subseteq u_2 \wedge u_2 \not\subseteq u_1$
    $\wedge \forall u \subsetneq u_1 : u \subsetneq u_2 \wedge \forall u \subsetneq u_2 : u \subsetneq u_1\}$
  $R_l \leftarrow (\subseteq \cup (R_\parallel \cap R_h))^+$ ▷ order $R_\parallel$ via R_h,
    ▷ $^+$ denotes the reflexive-transitive closure
  $T_n \leftarrow$ *enumerate*(T, R_l) ▷ set $T \rightarrow$ sequence T_n
query *get* $(T \subseteq U) : M \in \mathbb{M}$
  $T_n \leftarrow$ *linearize*$(T^{\downarrow U})$
  $M \leftarrow T_0 \uplus T_1 \uplus \ldots \uplus T_n$

3.2 Outlook on Systemic Access Control

Access control is usually enforced by a centralized entity in a strongly consistent way. In a distributed, weakly-consistent setting, we have to embrace that time is only a partial ordering, events and administrative changes happen concurrently, and there is no such thing as consensus on a linear order of events or on which policies are in in effect 'now' [15].

Previous works on access control for CRDTs mainly focus on filesystem-like cryptographically-enforceable access control in closed groups [13, 17], while our outlook is inspired by the granular authorizations and administrative permissions of Matrix [4] and similar systems [2].

The idea is to provide *systemic* access control, i.e., storing attributes needed for policies as well as policies itself in the types, thereby allowing integration of access decisions in the CRDT functions and gaining a decentralized enforcement. In contrast with the bare EDP formalizations from Section 2.2, decentralized enforcement requires that a replica can prove to another replica that a third replica was responsible for an update, and thereby requires transferable authentication through digital signatures for the update creator. Such a systemic access control can be constructed by a composition of a CEDP – to gain a concept of logical time through the happened-after relation – and multiple EPMs.

The main challenge is to treat concurrent administrative changes securely. To deal with concurrent, conflicting changes, the key idea is that a concurrent update is never *rejected* if it was authorized for its downward closure, it just might be ignored on linearization if another change wins. A prototype model of such a composed data type is defined and discussed in the appendix (supplemental material), together with a proof sketch of the CRDT property.

4 Conclusion & Future Work

Based on a formalization using partially-ordered sets, we presented Extend-only Directed Posets (EDPs) as unifying generalization of ideas around DAG CRDTs in Byzantine environments. The state-based formalization shows the essence of these CRDTs, while the operation-based formalization makes the optimization for efficiency explicit. Based on EDPs, one can derive modular building blocks and compose complex Byzantine-tolerant CRDTs with ease, as exemplified by the map data type and indicated by the use case for systemic decentralized access control. Using composition of these building blocks, a formalization of the access-control-included CRDT of the popular Matrix messaging system can be gained. We outline corresponding models and proofs in the supplemental material of this paper. This work-in-progress paper, therefore, serves as a step towards formal verification of strong eventual consistency in Byzantine environments of both EDP-based system designs as well as corresponding implementations (like Matrix), including security properties of decentralized access control.

References

[1] Luca Bombelli, Joohan Lee, David Meyer, and Rafael D. Sorkin. 1987. Space-time as a causal set. *Phys. Rev. Lett.* 59 (Aug 1987), 521–524. Issue 5. https://doi.org/10.1103/PhysRevLett.59.521

[2] Herb Caudill. 2023. *Local First Auth: Decentralized authentication and authorization for team collaboration, using a secure chain of cryptological signatures.* local-first-web. https://github.com/local-first-web/auth

[3] Vicent Cholvi, Antonio Fernández Anta, Chryssis Georgiou, Nicolas Nicolaou, Michel Raynal, and Antonio Russo. 2021. Byzantine-tolerant Distributed Grow-only Sets: Specification and Applications. arXiv:2103.08936 [cs.DC]

[4] Matrix Specification Contributors. 2023. *Matrix specification v1.6.* Technical Report. The Matrix.org Foundation C.I.C. https://spec.matrix.org/v1.6/.

[5] Florian Jacob, Saskia Bayreuther, and Hannes Hartenstein. 2022. On CRDTs in Byzantine Environments: Conflict Freedom, Equivocation Tolerance, and the Matrix Replicated Data Type. In *Sicherheit 2022 (GI-Edition: Proceedings, Vol. 323)*. Gesellschaft für Informatik (GI), Bonn, 113–128. https://doi.org/10.18420/sicherheit2022_07

[6] Florian Jacob, Luca Becker, Jan Grashöfer, and Hannes Hartenstein. 2020. Matrix Decomposition – Analysis of an Access Control Approach on Transaction-based DAGs without Finality. In *Proceedings of the 25th ACM Symposium on Access Control Models and Technologies* (Barcelona, Spain) *(SACMAT '20)*. Association for Computing Machinery, New York, NY, USA, 81–92. https://doi.org/10.1145/3381991.3395399

[7] Florian Jacob, Carolin Beer, Norbert Henze, and Hannes Hartenstein. 2021. Analysis of the Matrix Event Graph Replicated Data Type. *IEEE access* 9 (2021), 28317–28333. https://doi.org/10.1109/ACCESS.2021.3058576

[8] Martin Kleppmann. 2022. Making CRDTs Byzantine Fault Tolerant. In *Proceedings of the 9th Workshop on Principles and Practice of Consistency for Distributed Data* (Rennes, France) *(PaPoC '22)*. Association for Computing Machinery, New York, NY, USA, 8–15. https://doi.org/10.1145/3517209.3524042

[9] Martin Kleppmann and Heidi Howard. 2020. Byzantine eventual consistency and the fundamental limits of peer-to-peer databases. arXiv:2012.00472 [cs.DC]

[10] Leslie Lamport. 2019. *Time, Clocks, and the Ordering of Events in a Distributed System.* Association for Computing Machinery, New York, NY, USA, 179–196. https://doi.org/10.1145/3335772.3335934

[11] Ranganathan Padmanabhan and Sergiu Rudeanu. 2008. *Axioms for lattices and Boolean algebras.* World Scientific, Singapore. https://doi.org/10.1142/7007

[12] Nuno Preguiça, Carlos Baquero, and Marc Shapiro. 2018. Conflict-free replicated data types (CRDTs). arXiv:1805.06358 [cs.DC]

[13] Pierre-Antoine Rault, Claudia-Lavinia Ignat, and Olivier Perrin. 2022. Distributed Access Control for Collaborative Applications Using CRDTs. In *Proceedings of the 9th Workshop on Principles and Practice of Consistency for Distributed Data* (Rennes, France) *(PaPoC '22)*. Association for Computing Machinery, New York, NY, USA, 33–38. https://doi.org/10.1145/3517209.3524826

[14] Marc Shapiro, Nuno Preguiça, Carlos Baquero, and Marek Zawirski. 2011. Conflict-Free Replicated Data Types. In *Stabilization, Safety, and Security of Distributed Systems*, Xavier Défago, Franck Petit, and Vincent Villain (Eds.). Springer, Berlin, Heidelberg, 386–400. https://doi.org/10.1007/978-3-642-24550-3_29

[15] Mathias Weber, Annette Bieniusa, and Arnd Poetzsch-Heffter. 2016. Access Control for Weakly Consistent Replicated Information Systems. In *Security and Trust Management*, Gilles Barthe, Evangelos Markatos, and Pierangela Samarati (Eds.). Springer International Publishing, Cham, 82–97. https://doi.org/10.1007/978-3-319-46598-2_6

[16] Matthew Weidner, Heather Miller, and Christopher Meiklejohn. 2020. Composing and Decomposing Op-Based CRDTs with Semidirect Products. *Proc. ACM Program. Lang.* 4, ICFP, Article 94 (2020), 27 pages. https://doi.org/10.1145/3408976

[17] Elena Yanakieva, Michael Youssef, Ahmad Hussein Rezae, and Annette Bieniusa. 2021. Access Control Conflict Resolution in Distributed File Systems Using CRDTs. In *Proceedings of the 8th Workshop on Principles and Practice of Consistency for Distributed Data* (Online, United Kingdom) *(PaPoC '21)*. Association for Computing Machinery, New York, NY, USA, Article 1, 3 pages. https://doi.org/10.1145/3447865.3457970