Coinductive Reasoning about CRDT Emulation

Nathan Liittschwager[1], Stelios Tsampas[2],
Jonathan Castello[1], and Lindsey Kuper[1]

[1] University of California, Santa Cruz, USA
[2] Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Abstract. Conflict-free replicated data types (CRDTs) are distributed data structures designed for fault tolerance and high availability. CRDTs have historically been taxonomized into operation-based (or op-based) CRDTs and state-based CRDTs. The notion that state-based and op-based CRDTs are equivalent is often appealed to in the literature. In particular, verification techniques and results for one kind of CRDT are often said to be applicable to the other kind, thanks to this equivalence. However, while there are general algorithms for constructing a state-based CRDT from a given op-based CRDT and vice versa, what it means for one kind of CRDT to emulate another has never been made fully precise. In this paper, we model CRDT systems as transition system coalgebras, and argue that emulation can be understood formally in terms of various *weak simulations* between the coalgebras of the original and emulating CRDT systems. As a simple corollary, we deduce which properties are preserved by the emulation algorithms, thus closing a gap in the CRDT literature.

Keywords: CRDTs, coalgebras

1 Introduction

In distributed data storage systems, *data replication* is a ubiquitous mechanism for guarding against machine failures and ensuring that data is physically close to far-flung clients. Informally, replication takes an object and copies it over n sites, or *nodes*, with each *replica* acting as an independent copy of the original object. With replication comes the challenge of ensuring that replicas remain (more or less) consistent with one another, especially in the face of inevitable network partitions and clients who demand "always-on" access to data. The ideal of consistency for such a replicated system is *linearizability* [18], under which clients cannot tell whether they are interfacing with the original object, or the replicated system. Unfortunately, linearizability is impractical to implement — indeed, systems that prioritize *high availability* of data must necessarily do so at the expense of linearizability [13, 14].

The quest for an optimal point in the availability/consistency trade-off space has led to the development of *conflict-free replicated data types* (CRDTs) [43, 35, 34], which are data structures designed for replication and high availability.

CRDTs sacrifice linearizability in favor of a weaker *strong convergence* property [43], which says that replicas that have received and applied the same *set* of updates will agree in state, regardless of the order in which those updates were received and applied. When coupled with a guarantee of *eventual delivery* of updates to replicas, strong convergence ensures that replicas will *eventually* come to agree. In the last decade, CRDTs have been an active area of research in programming languages and verification communities, with considerable attention paid to the formal specification and verification (of various properties, but especially strong convergence) of CRDT designs [9, 45, 15, 12, 25, 31, 24, 32, 33], with recent work moving toward automated verification [30, 10] and even synthesis of correct-by-construction CRDTs [22].

In their pioneering work on CRDTs, Shapiro et al. [43] taxonomize CRDTs into *state-based* CRDTs, in which replicas apply updates locally and periodically broadcast their local state to other replicas over the network, and *operation-based* (or *op-based*) CRDTs, in which every state-updating operation is broadcast and applied at each replica.[3] In state-based CRDTs, the states that a replica can take on must be elements of a join-semilattice, and a replica receiving an update from a remote replica will apply the update by taking the least upper bound (join) of its local state and the received update. Op-based CRDTs, on the other hand, only require concurrent operations to commute, but rely on stronger ordering guarantees (in particular, *causal broadcast* [8, 7]) from the underlying network transport mechanism. Both the state-based and op-based approaches result in strong convergence, the defining characteristic of CRDTs.

Most work on CRDT specification and verification focuses on either state-based [45, 12, 31, 44, 33, 22] or op-based [15, 30, 25, 24, 32] CRDTs exclusively. The justification for this choice is that state-based and op-based CRDTs can *emulate* each other. Shapiro et al. [43] give general algorithms by which one may construct a state-based CRDT out of a given op-based CRDT, and vice versa. However, Shapiro et al. stop short of formally defining a notion of emulation and proving that their construction satisfies it. Yet the notion that state-based and op-based CRDTs can emulate each other is frequently appealed to in the literature. For instance, Nagar and Jagannathan [30], in their work on verification of op-based CRDTs, write that "our technique naturally extends to state-based CRDTs since they can be emulated by an op-based model," and Laddad et al. [22], in their work on synthesis of state-based CRDTs, write that they "can always be translated to op-based CRDTs if necessary." Hence this emulation notion is "load-bearing" and therefore deserving of being made precise.

In this paper, we seek to close this gap in the CRDT literature and formalize the notion of emulation. To that end, we give a simple transition system model of CRDT systems, modeling the network and interactions between replicas. This model can be understood from the perspective of *universal coalgebra* [37, 21]. Given an endofunctor $F\colon \mathcal{C} \to \mathcal{C}$ in some category $\mathcal{C}$, F-coalgebras are pairs

[3] More recently, Almeida et al. [3] introduce *delta state* CRDTs, an optimization of traditional state-based CRDTs in which only state changes, rather than entire states, must be disseminated over the network.

(X, h) of an object $X \in \mathcal{C}$ and a morphism $h\colon X \to FX$. The idea is that the endofunctor F encapsulates the generic behavior of a system. This simple abstraction has proven to be remarkably general, capable of modelling systems such as deterministic and non-deterministic automata, Mealy machines [36], probabilistic systems [5] and higher-order languages [16] by varying the choice of category $\mathcal{C}$ and endofunctor F, yet powerful enough to be the foundation of a unifying theory of bisimulation and coinduction. It is through this unifying theory of coalgebra that we can reason coinductively about emulation of CRDTs.

Our contributions are (1) to demonstrate that the operational semantics of a CRDT system can be modeled by an appropriate transition system (and, therefore, by a coalgebra), and (2) to formalize precisely what is meant by *emulation* of CRDTs in terms of *weak simulations* between coalgebras. Our results give researchers working on CRDTs a rigorous way to think about equivalence of state-based and op-based CRDTs: in particular, properties that transfer from one kind of CRDT to the other are precisely those properties that are preserved by weak simulation relations.

The rest of this paper is organized as follows. After giving background on CRDTs (Section 2), we present a semantics for a system of CRDT replicas and make the straightforward observation that our semantics is a particular kind of coalgebra (Section 3). We then review the formal notion of simulations in our setting, and argue how it can be adapted to characterize *emulation* (Section 4). We justify this choice of simulation by reasoning that emulation should be about relating the observable behaviors of the original system and the emulating system. We have kept the emulation algorithms of Shapiro et al. intact, albeit with minor modifications. This results in two emulation *mappings*, one in each direction of emulation. Each emulation mapping corresponds to *two* simulation arguments, which we argue coinductively. We conclude with a discussion of related work (Section 5).

2 Background on CRDTs

In this section, we recall the definitions of op-based and state-based CRDTs, based on those of Shapiro et al. [43]. CRDTs are networks of communicating replicas, but the definitions of Shapiro et al. specify only the *interface* that each individual replica exposes to the network, as this fully determines the semantics, i.e., the behavior, of the CRDT network as a whole. Here, what we refer to as a *system* or a *network* is a collection of non-byzantine processes, which we call *replicas*, that communicate with each other through an underlying asynchronous communication protocol. Later, in Section 3, we will recast these definitions in terms of coalgebras.

CRDTs are constructed in such a way as to guarantee *strong eventual consistency* [43, §2.2] of replicas, the most salient aspect which is of *strong convergence*, which says that replicas that have received the same (unordered) set of updates have equivalent state.

Assumption 1. In what follows, we fix a set A of *commands* and a set B of *observables*. We also assume that CRDT systems consist of $n \in \mathbb{N}$ replicas.

2.1 Op-based CRDTs

The definition below formalizes operation-based CRDTs, originally introduced by Shapiro et al. [43, §2.4]. We refer to them simply as *op-based CRDTs*.

Definition 2 (Op-based CRDT)**.** An *op-based CRDT* is a tuple $(S, s_0, M, \mathtt{u}, \mathtt{t}, \mathtt{e}, \mathtt{q})$, consisting of:

- A set S of *local states*.
- An initial state $s_0 : S$.
- A set M of *messages*.
- An *update* map $\mathtt{u} : S \times A \to S$.
- A *prepare-update* map $\mathtt{t} : S \times A \to M$.
- An *effect-update* map $\mathtt{e} : S \times M \to S$.
- A *query* map $\mathtt{q} : S \to B$.

The core principle behind op-based CRDTs is that replicas locally (and independently of the other replicas) execute commands, which are then propagated to the rest of the network via broadcast messages. The mechanism works as follows: suppose a given replica i, with a local state of $s_i \in S$, executes a command $a \in A$, thus transitioning to the state $s_i' = \mathtt{u}(s_i, a)$. Replica i then prepares a message $m = \mathtt{t}(s_i, a)$, which is broadcast to all other replicas. The broadcast happens asynchronously, meaning messages are received by other replicas in an indeterminate order. Finally, the replicas may "consume" the message m via the effect-update map $\mathtt{e}$, e.g., replica j applies $\mathtt{e}(s_j, m)$. In this case, we say that the message m is *delivered* at replica j. The query function $\mathtt{q}$ exposes the observable part of a replica's local state to the external world.

Notation 3. Often we will consider applying *sequences* of messages. The following notation is thus useful. Let M^* denote the set of strings of messages (where $(-)^*$ Kleene star). Let ε be the empty string and $\cdot$ the usual string concatenation. $\forall w \in M^*$, define $\mathtt{e}_w : S \to S$ inductively by setting $\mathtt{e}_\varepsilon(s) = s$, and $\mathtt{e}_{m \cdot w}(s) = \mathtt{e}_w(\mathtt{e}(s, m))$ for all $m \in M$.

Remark 4 (The role of causality in op-based CRDTs)**.** An important detail about op-based CRDTs is that they must be implemented on top of a *causal broadcast* mechanism. Causal broadcast ensures that when a broadcast message m is delivered at a replica j, any message sent before m (in the specific sense of Lamport's *happens-before* partial order [23]) will have already been delivered at j. (Messages not ordered by the happens-before relation, on the other hand, may be delivered in arbitrary order.) Protocols for causal message delivery are well known in the distributed systems literature [6, 39, 8, 7].

We express the causal broadcast requirement as follows: if M is the set of all messages in a given execution of a CRDT system, then $(M, \prec)$ is a partial order, and $\prec$ is called a *causality relation*, which encodes Lamport's happens-before relation. Intuitively, if $m' \prec m$, then the sending of message m' is a potential "cause" for the sending of message m. Furthermore, we say that m and m' are *concurrent* (written $m \parallel m'$) if $\neg(m \prec m') \wedge \neg(m' \prec m)$. As an axiom, op-based CRDTs require that the effects of concurrent messages *commute*. That is, in Notation 3,

$$\forall s \in S.\, m \parallel m' \implies \mathtt{e}_{m \cdot m'}(s) = \mathtt{e}_{m' \cdot m}(s) \tag{1}$$

Example 5. A typical example of an op-based CRDT is the *grow-only set* or *G-Set*. Fix a set of objects Σ, and define the local states as elements $s \in \mathcal{P}(\Sigma)$. Initially, $s_0 = \varnothing$ and there is only one type of command: $\mathtt{add}(e)$ where $e \in \Sigma$. Then we define

- *prepare-update*: $\mathtt{t}(s, \mathtt{add}(e)) = (\mathtt{add}, e)$
- *effect-update*: $\mathtt{e}(s, (\mathtt{add}, e)) = s \cup \{e\}$
- *update*: $\mathtt{u}(s, \mathtt{add}(e)) = \mathtt{e}(s, \mathtt{t}(s, \mathtt{add}(e)))$

Queries enable clients to check if elements are in the set, meaning $B = 2^{\Sigma}$ and $q : \mathcal{P}(\Sigma) \cong 2^{\Sigma}$. Furthermore, all updates commute, so causal delivery is in fact not necessary for this particular CRDT. However, more sophisticated op-based CRDTs such as *add-remove sets*, do require causal broadcast to ensure strong convergence.

2.2 State-based CRDTs

Next, we define state-based CRDTs [43, §2.3], in which local replica states form a join-semilattice.

Definition 6 (State-based CRDT)**.** A *state-based CRDT* is a tuple $((S, \sqcup), s_0, \mathtt{u}, \mathtt{q})$, consisting of:

- A join-semilattice $(S, \sqcup)$.
- An initial state $s_0 : S$.
- An *update* map $\mathtt{u} : S \to S^A$ that is furthermore *inflationary*, i.e.

$$s \leq \mathtt{u}(s, a) \text{ for all } s \in S \text{ and } a \in A.$$

- A *query* map $\mathtt{q} : S \to B$.

As is standard, we write $x \leq y$ whenever $x \sqcup y = y$.

A state-based CRDT operates as follows: as in op-based CRDTs, replicas may locally and independently execute commands $a \in A$. The synchronization mechanism, however, is radically different; whereas an op-based CRDT system uses a causal broadcast mechanism, in a state-based CRDT system any replica i may initiate a synchronization exchange with another replica j. In this exchange, replica i sends its entire local state s_i to replica j through the network. Upon arrival of the message (with payload s_i), replica j joins s_i with its current state s_j, i.e. $s'_j = s_i \sqcup s_j$. In that case, we say that (the message with payload) s_i is *delivered* at replica j.

Example 7. One kind of state-based CRDT with many applications in distributed systems is a *Lamport clock* [23]. The join-semilattice is $(\mathbb{N}, max : \mathbb{N} \times \mathbb{N} \to \mathbb{N})$, where *max* takes its usual meaning. The initial state is $\texttt{clock}_0 = 0$. The command is simply `tick`, and the *update* is defined $\texttt{u}(\texttt{clock}, \texttt{tick}) = \texttt{clock} + 1$. Queries compare a given clock to the current `clock` that is the replica's internal state. Then n replicas are seen as n distributed logical clocks, and may be used to track the order of events in a distributed system.

3 CRDTs, coalgebraically

In this section, we formalize the behavior of CRDTs as *non-deterministic transition systems* on vectors of replica states. The following definition of a replica state covers both state-based and op-based CRDTs.

Notation 8. We write $\mathcal{M} : \mathbf{Set} \to \mathbf{Set}$ for the multiset functor.

Definition 9 (Replica states)**.** Let M and S be sets, representing resp. *messages* and *local states*. A *replica state* is a pair $(s, \sigma) \in S \times \mathcal{M}(M)$.

A replica state (s, σ) represents the runtime information of a single replica, which consists of the local state s and its *message buffer* σ. A vector of replica states thus represents the runtime information of a CRDT system; a network of replicas. The following definition captures the nature of a CRDT system and is independent of whether it is state-based or op-based.

Definition 10. Global State and Configurations Let $\mathcal{E}$ be a set of events and $S \times \mathcal{M}(M)$ the set of replica states. We say a tuple $(x_i)_{i \in n} \in (S \times \mathcal{M}(M))^n$ is a *global state* of n replicas and $\langle \alpha, (x_i)_{i \in n} \rangle \in \mathcal{E} \times (S \times \mathcal{M}(M))^n$ is an *augmented global state* or *configuration*.

Definition 11 (CRDT system)**.** A *CRDT system* (on n replicas) is a pair (R, obs) consisting of a query map $\mathsf{obs} : S \to B$ and a transition relation

$$R \subseteq \mathcal{E} \times (S \times \mathcal{M}(M))^n \times \mathcal{E} \times (S \times \mathcal{M}(M))^n).$$

Remark 12. CRDT systems form coalgebras in the category **Set** of sets and total functions. In particular, a CRDT system (R, obs) is equivalently the coalgebra

$$(h, \vec{\mathsf{obs}}) : X \to \mathcal{P}(X) \times B^n$$

where $X = \mathcal{E} \times (S \times \mathcal{M}(M))^n$, $\vec{\mathsf{obs}} = (\mathsf{proj}_2 \circ \mathsf{obs})^n$ and $\vec{y} \in h(\vec{x})$ if $R(\vec{x}, \vec{y})$.

From the coalgebraic perspective, the state space of CRDT systems corresponds to an n-sized vector of replica states, tagged with event-labels; the observables are collected by the map $\mathsf{obs} : S \to B$, acting on the local state of each replica. Recall that our CRDTs are always equipped with such a map.

Next, for both op-based and state-based CRDTs, we describe the semantics of CRDT systems, including how a local replica changes state, and how replicas interact with each other.

3.1 Op-based CRDT system semantics

Assumption 13. For the purposes of Section 3.1, we assume an op-based CRDT $(S, s_0, M, \mathtt{u}, \mathtt{t}, \mathtt{e}, \mathtt{q})$.

We model the behavior of individual replicas in an op-based CRDT using a transition relation $\longrightarrow_{\mathsf{op}}$ on replica states. For the op-based CRDT in Assumption 13, we pick the set of messages in replica states to be M. Specifically for op-based CRDTs, commands $a \in A$ also generate a *message* $m = \mathtt{t}(s, a) \in M$ to be broadcast to other replicas. We denote a step that generates message m via command a with the notation $x_i \longrightarrow_{\mathsf{op}} x'_i \uparrow (a, m)$. The replica semantics of the op-based CRDT is given by the following rules:

$$
\frac{s' = \mathtt{u}(s,a) \quad m = \mathtt{t}(s,a)}{(s,\sigma) \longrightarrow_{\mathsf{op}} (s',\sigma) \uparrow (a,m)} \ [\mathsf{OpLUpd}]
\qquad
\frac{\exists m \in \sigma \quad \mathsf{deliverable}(s,m) = \top \quad s' = \mathtt{e}(s,m)}{(s,\sigma) \longrightarrow_{\mathsf{op}} (s', \sigma \setminus \{m\})} \ [\mathsf{OpLRecv}] \tag{2}
$$

In rule $\mathsf{OpLStep}$, the replica executes a command $a \in A$, generating message m and changing the replica's state accordingly. In $\mathsf{OpLRecv}$, the replica in question *delivers* a message $m \in \sigma$, i.e. it executes the underlying command included in the message ($\mathtt{emap}(s, m)$) and removes the message from its message buffer. We defer discussion of the $\mathsf{deliverable}(s, m) = \top$ until Remark 16.

With the above replica semantics in mind, the *global* semantics of an op-based CRDT system models communication between replicas, with computation modeled by a transition relation on states augmented with *events* or *actions*, denoting what sort of computation step took place. More formally, if x, x' are global states and α, α' are events, then write $\langle \alpha, x\rangle \rightsquigarrow_{\mathsf{op}} \langle \alpha', x'\rangle$ to mean that x transitions to x' via event α'.

To be precise, we now define the events in op-based CRDT systems.

Definition 14. For $i, j \in n$, $m \in M$ and $a \in A$,

$$
\mathcal{E}_{\mathsf{op}} \ni \alpha ::= \top \mid \mathtt{upd}^i(a,m) \mid \mathtt{bc}^i(m) \mid \mathtt{dlvr}^{j \leftarrow i}(m) \tag{3}
$$

where $\top$ is a special *null* event (for initial states), $\mathtt{upd}^i(a, m)$ denotes a local update at replica i with command a and emitting message m, $\mathtt{bc}^i(m)$ denotes the sending of m from replicas i to all other other replicas $j \in n$, and $\mathtt{dlvr}^{j \leftarrow i}(m)$ denotes the delivery of message m at replica j, where the sender is replica i.

Write $\alpha \notin \mathtt{update}$ to mean α is not an update event. Then we define the global transition relation $\rightsquigarrow_{\mathsf{op}}$ as follows:

$$
\frac{\alpha \notin \mathtt{update} \quad x_i \longrightarrow_{\mathsf{op}} x'_i \uparrow (a,m)}{\langle \alpha, (x_1, .., x_n)\rangle \rightsquigarrow_{\mathsf{op}} \langle \mathtt{upd}^i(a,m), (x_1, ..., x'_i, ..., x_n)\rangle} \ [\mathsf{OpUpdate}]
$$
$$
\frac{(x'_1, ..., x'_n) = \mathtt{bcast}^i_m(x_1, ..., x_n)}{\langle \mathtt{upd}^i(a,m), (x_1, ..., x_n)\rangle \rightsquigarrow_{\mathsf{op}} \langle \mathtt{bc}^i(m), (x'_1, \ldots, x'_n)\rangle} \ [\mathsf{OpBroadcast}] \tag{4}
$$
$$
\frac{\alpha \notin \mathtt{update} \quad x_i \longrightarrow_{\mathsf{op}} x'_i}{\langle \alpha, (x_1, ..., x_n)\rangle \rightsquigarrow_{\mathsf{op}} \langle \mathtt{dlvr}^{j \leftarrow i}(m), (x_1, ..., x'_i, ..., x_n)\rangle} \ [\mathsf{OpRecv}]
$$

In our model, transitions taken by each replica can be seen as atomic, local computations, and in a CRDT network, replicas perform local computations in no particular order. As such, rule OpRecv acknowledges that if a replica can take a step individually (by delivering a message via OpLRecv), it can also do so inside the network. Local updates at a replica induce a cascade of global transitions: when a replica updates its state via command a, it also does so inside the network. At the same time, the replica prepares a message m to be emitted to the network, and then does so immediately. This is captured by the OpUpdate and OpBroadcast rules in the global semantics. Note that the premise $\alpha \notin \texttt{send}$ in OpUpdate and OpRecv requires that broadcast events always follow an update event. A broadcast $\mathsf{bcast}^i_m(x_1, \dots, x_n)$ returns the global state $(x'_1, ...x'_n)$, where message m is placed in the message buffer of each replica j such that $j \neq i$. We stress that this communication is asynchronous: a message being placed in some replica's message buffer denotes that the message has simply been *sent*. Asynchrony is modeled by the fact that there can be an arbitrary number of steps between the sending/reception and the *delivery* (i.e., consumption) of the message.

Definition 15. Let $(x_i)_{i\in n}$ be a global state and α an event. We say a (possibly empty) sequence $\langle \alpha_i, z^i\rangle_{i\in k}$ of configurations is a *trace* of $\langle \alpha, (x_i)_{i\in n}\rangle$ if

$$(\langle \alpha, (x_i)_{i\in n}\rangle \rightsquigarrow_{\mathsf{op}} \langle \alpha_1, z^1\rangle) \wedge (\forall i \geq 1.\, \langle \alpha_i, z^i\rangle \rightsquigarrow_{\mathsf{op}} \langle \alpha_{i+1}, z^{i+1}\rangle).$$

If all the events $(\alpha_i)_{i\in k}$ have a certain form (e.g., $\alpha_i = \texttt{dlvr}^{j_i \leftarrow l_i}(m_i)$) we may speak of *traces of events.*

Remark 16. We now turn to a more precise discussion of the $\mathsf{deliverable}$ premise in the OpLRecv rule. Recall from Remark 4 that op-based CRDTs are built on top of causal broadcast, and that the semantics of a causal broadcast mechanism induces a causality relation $\prec$ where *(i.)* $(M, \prec)$ is a partial order, and *(ii.)* causal broadcast guarantees that messages delivered at each replica are delivered in an order consistent with $\prec$.

We now show how such a relation $\prec$ might be obtained in our semantics. Let $\langle \alpha_i, z^i\rangle_{i\in k}$ be a trace of initial state $\langle \top, (s^0, \varnothing)_{i\in n}\rangle$ (Definition 15), and take the set of events $E = \bigcup_{i\in k}\{\alpha_i\}$, which may be partitioned as $E = \bigcup_{i\in n} E_i$ by binning events α_i based on where they occured (e.g., E_j includes all events of the form $\texttt{upd}^j(a)$, $\texttt{bc}^j(m)$ and $\texttt{dlvr}^{j\leftarrow i'}(m)$).

We then define a standard [23] happens-before relation $\prec_{\mathsf{hb}}$ as the smallest relation on events E satisfying

- $p < q$ and α_p and α_q are both events on the same replica (e.g., $\alpha_p, \alpha_q \in E_j$)
- $\alpha_p = \texttt{bc}^i(m)$ and $\alpha_q = \texttt{dlvr}^{j\leftarrow i}(m)$; and
- $\exists \alpha_r$ event s.t. $\alpha_p \prec_{\mathsf{hb}} \alpha_r$ and $\alpha_r \prec_{\mathsf{hb}} \alpha_q$.

The relation $\prec_{\mathsf{hb}}$ is a partial order relation and induces the partial order relation $\prec$ on messages: $m' \prec m \iff \texttt{bc}^i(m') \prec_{\mathsf{hb}} \texttt{bc}^j(m)$. We can understand the partial order $(M, \prec)$ induced by the causal broadcast mechanism as essentially the abstraction of this construction.

A standard implementation strategy for such a mechanism [6, 7, 8] is to have the sender of a message augment the message with causal metadata (for instance, a *vector clock* [27, 11, 40]) that summarizes information about the causally preceding messages, and have each replica additionally maintain causal metadata as part of its state.

It is then the job of the causal broadcast mechanism to enforce *causal delivery*: each $\mathtt{dlvr}^{j \leftarrow i}(m)$ event can only occur if all messages $m' \prec m$ in the execution have already been delivered to replica j. This is done by inspecting the message's causal metadata and comparing it with the causal metadata in the replica's state to determine whether the message can be safely delivered or needs to be buffered for later delivery. We leave the details of the causal delivery enforcement mechanism abstract, hiding all causal metadata, and instead capture its semantics using the $\mathsf{deliverable}$ relation and the partial order $(M, \prec)$.

In particular, $\mathsf{deliverable}(s, m) = \top$ in the premise of $\mathsf{OpLRecv}$ means that all messages m' such that $m' \prec m$ have already been delivered at the replica in question.

Remark 17. The replica and global semantics of the op-based CRDT, coupled with the query map $\mathtt{q} : S \to B$, forms the CRDT system $(\rightsquigarrow_{\mathsf{op}}, \mathtt{q})$ in the sense of Definition 11.

3.2 State-based CRDT semantics

Assumption 18. For the purposes of Section 3.2, we assume a state-based CRDT $((S, \sqcup), s_0, \mathtt{u}, \mathtt{q})$.

We proceed analogously to the op-based CRDT case. For the state-based CRDT $((S, \sqcup), s_0, \mathtt{u}, \mathtt{q})$, we pick the set of messages in replica states to be S, as we are sending entire internal states to other replicas. The replica semantics is given by the following rules:

$$\begin{gathered}
\frac{a \in A \quad s' = \mathtt{u}(s, a)}{(s, \sigma) \longrightarrow_{\mathsf{st}} (s', \sigma)} \ [\mathsf{StLUpd}] \\
\frac{\exists s' \in \sigma}{(s, \sigma) \longrightarrow_{\mathsf{st}} (s \sqcup s', \sigma \setminus \{s'\})} \ [\mathsf{StLRecv}]
\end{gathered} \tag{5}$$

Compared to the op-based case, the semantics are quite similar at a local level, only now replicas are not required to broadcast as a consequence of an update. Delivering a message from another replica is just a join operation. Sending a state to another replica is given as a separate step in the global semantics, which are given below over the following events.

Definition 19. For $i, j \in n$, and $s \in S$, and $a \in A$,

$$\mathcal{E}_{\mathsf{st}} \ni \alpha = \top \mid \mathtt{upd}^i(a) \mid \mathtt{send}^{i \to j}(s) \mid \mathtt{dlvr}^{j \leftarrow i}(s) \tag{6}$$

where the events are analogous to those in Definition 14, with the caveat that now send events are point-to-point rather than broadcasts. Now, for events $\alpha \in \mathcal{E}_{\mathsf{st}}$, we have the rules:

$$
\frac{x_i = (s, \sigma) \quad x_i \longrightarrow_{\mathsf{st}} x_i' \quad x_i' = (\mathtt{u}(s, a), \sigma)}{\langle \alpha, (x_1, \ldots, x_i, \ldots, x_n) \rangle \rightsquigarrow_{\mathsf{st}} \langle \mathtt{upd}^i(a), (x_1, \ldots, x_i', \ldots, x_n) \rangle} \ [\mathsf{StUpdate}]
$$

$$
\frac{x_i = (s_i, \sigma_i) \quad x_j = (s_j, \sigma_j) \quad x_j' = (s_j, \sigma_j \cup \{s_i\})}{\langle \alpha, (x_1, ..., x_j, ..., x_n) \rangle \rightsquigarrow_{\mathsf{st}} \langle \mathtt{send}^{i \to j}(s_i), (x_1, ..., x_j', ..., x_n) \rangle} \ [\mathsf{StSend}] \tag{7}
$$

$$
\frac{x_j = (s_j, \sigma) \quad x_j \longrightarrow_{\mathsf{st}} x_j' \quad x_j' = (s_j \sqcup s, \sigma \setminus \{s\})}{\langle \alpha, (x_1, .., x_j, ..., x_n) \rangle \rightsquigarrow_{\mathsf{st}} \langle \mathtt{dlvr}^{j \leftarrow i}(s), (x_1, ..., x_j', ..., x_n) \rangle} \ [\mathsf{StRecv}]
$$

The global semantics for state-based CRDT systems are interpreted similarly to op-based CRDT systems. The critical difference is that while op-based systems are required to broadcast messages as a result of a local update, state-based systems need no requirement. The rules StUpdate and StRecv show that that replicas which take a step may also take a step inside the network. The rule StSend models a spontaneous synchronization connection initiated by replica i towards replica j with $i \neq j$. As we can see from the premise, the internal state of x_i is placed in the message buffer of replica j. Again, the communication is asynchronous, as replica j may consume this message at an arbitrary point in the future.

Remark 20. As with op-based CRDTs, the replica and global semantics of the state-based CRDT, coupled with the query map $\mathtt{q} : S \to B$, forms the CRDT system $(\rightsquigarrow_{\mathsf{st}}, \mathtt{q})$ in the sense of Definition 11.

Remark 21. The reason for augmenting the state with events, rather than using a labeled transition (i.e., using behavior functor $\mathcal{P}(\mathcal{E} \times -)$) is because the event types $\mathcal{E}_{\mathsf{op}}$ and $\mathcal{E}_{\mathsf{st}}$ in the op-based rules in Equation (4) and state-based rules in Equation (7) are not identical up to relabeling. In the weak simulation arguments we will present in Section 4, a single labeled event must be simulated by *multiple* other labeled events. To this end, we augment the states with events, thus keeping the behavior functor $\mathcal{P}$ in both systems. Passing to the following intermediate transition system for $(\rightsquigarrow_{\mathsf{st}}, \mathtt{q}_{\mathsf{st}})$-systems is useful for relating events $\mathcal{E}_{\mathsf{op}}$ to those in $\mathcal{E}_{\mathsf{st}}$.

Definition 22. Set $\mathtt{q}' = \mathtt{q}_{\mathsf{st}}$ and let $\mathcal{E}$ be the set of events given by the syntax

$$
\mathcal{E} \ni \alpha := \top \mid \mathtt{upd}^i(a) \mid \mathtt{bc}^i(s) \mid \mathtt{dlvr}^{j \leftarrow i}(s)
$$

where $s \in S$. Then construct a transition system $(\longmapsto, \mathtt{q}'')$ by defining $\forall \alpha$ events, $\forall z, z' \in (S \times \mathcal{M}(S))^n$ global states,

1. $\langle \alpha, z \rangle \longmapsto \langle \mathtt{upd}^i(a), z' \rangle \iff \exists \alpha'.\ \langle \alpha', z \rangle \rightsquigarrow_{\mathsf{st}} \langle \mathtt{upd}^i(a), z' \rangle$
2. $\langle \alpha, z \rangle \longmapsto \langle \mathtt{dlvr}^{j \leftarrow i}(s), z' \rangle \iff \exists \alpha'.\ \langle \alpha', z \rangle \rightsquigarrow_{\mathsf{st}} \langle \mathtt{dlvr}^{j \leftarrow i}(s), z' \rangle$.
3. $\langle \alpha, z \rangle \longmapsto \langle \mathtt{bc}^i(s), z' \rangle \iff \exists \alpha', \alpha_1, \ldots, \alpha_{n-1}$ events, $\exists z^1, \ldots, z^{n-1}$ states s.t.
$$
(\langle \alpha', z \rangle \rightsquigarrow_{\mathsf{st}} \langle \alpha_1, z^1 \rangle) \wedge (\forall j \geq 1.\ \langle \alpha_j, z^j \rangle \rightsquigarrow_{\mathsf{st}} \langle \alpha_{j+1}, z^{j+1} \rangle)
$$
where $z^{n-1} = z'$ and each $\alpha_j = \mathtt{send}^{i \to j}(s)$ s.t. $j \neq i$ ranges over n.

4 Emulation of CRDTs, coinductively

Shapiro et al. [43] argue that it is possible for an op-based CRDT to be *emulated* by a state-based CRDT and vice versa. To that end, they provide two transformations, one that constructs an op-based CRDT given a state-based CRDT, and one that constructs a state-based CRDT given an op-based CRDT. The translations are given in a precise manner, while the claim that op-based and state-based CRDTs emulate each other is left informal. In particular, there is no formal definition of "CRDT emulation"; rather, Shapiro et al. argue that their constructions preserve the *strong eventual consistency* (SEC) property of the original CRDTs. However, SEC preservation is insufficient for behavioral equivalence, since, for instance, a trivial CRDT that does nothing is also SEC.

To close this gap, we begin this section by first introducing our notion of CRDT emulation based on coalgebraic *(weak) simulation* (Section 4.1). Next, in Section 4.2, we show that Shapiro et al.'s translation of state-based CRDTs to op-based CRDTs is indeed an emulation in our sense, in that the resulting op-based system weakly simulates the state-based system and vice versa. We present the opposite direction in Section 4.3. It will become apparent that the two weak simulations are non-trivial and underline the differences between state-based and op-based CRDTs.

4.1 Emulation as coalgebraic simulation

We argue that the suitable notion to capture the relationship between op-based and state-based CRDTs is *(weak) simulation.* We move on to define simulations at the level of coalgebras for the endofunctor $B = L \times \mathcal{P}(-) : \mathbf{Set} \to \mathbf{Set}$, as our CRDT systems are all instances of B-coalgebras (recall Remark 12, Remark 17 and Remark 20).

Definition 23 (Simulations)**.** Let $(X, \langle \varepsilon, h\rangle)$ and $(Y, \langle \zeta, g\rangle)$ be coalgebras for endofunctor $B = L \times \mathcal{P}(-)$, for some set of observables L. A *simulation of* (X, h) *and* (Y, g) is a relation $R \subseteq X \times Y$ such that for all pairs of states $x \in X$, $y \in Y$ with $R(x, y)$, the following hold:

1. $\varepsilon(x) = \zeta(y)$.
2. If $x' \in h(x)$, then there exists $y' \in g(y)$ such that $R(x', y')$.

If $R(x, y)$, we say that y *simulates* x. Simulations are closed under arbitrary unions: if R and Q are simulations, so is $R \cup Q$. The greatest simulation thus exists and is the union of all simulations; this relation is known as *similarity*, written as $\lesssim$.

Definition 24 (Weak simulation)**.** Let $(X, \langle \varepsilon, h\rangle)$ and $(Y, \langle \zeta, g\rangle)$ be coalgebras for endofunctor $B = L \times \mathcal{P}(-)$, for some set of observables L, and let $g^* : Y \to \mathcal{P}(Y)$ be the *reflexive, transitive closure* of $g : Y \to \mathcal{P}(Y)$. That is, g^* is the least map $Y \to \mathcal{P}(Y)$ (w.r.t. pointwise inclusion), satisfying (i) $\forall y.\, g(y) \in g^*(y)$, (ii) $\forall y.\, y \in g^*(y)$ and (iii) $\forall y, y', y''.\, y' \in g^*(y) \wedge y'' \in g^*(y') \implies y'' \in g^*(y)$. A *weak simulation of* (X, h) *and* (Y, g) is a simulation of (X, h) and (Y, g^*).

Example 25. Let $(\longrightarrow_1, \mathsf{obs}_1)$ and $(\longrightarrow_2, \mathsf{obs}_2)$ be two CRDT systems according to Definition 11, and let $(\overrightarrow{\mathsf{obs}}_1, \longrightarrow_1) : X \to L \times \mathcal{P}X$ and $(\overrightarrow{\mathsf{obs}}_2, \longrightarrow_2) : Y \to L \times \mathcal{P}Y$ be the induced coalgebras from Remark 12. Instantiating Definition 24 to $(\overrightarrow{\mathsf{obs}}_1, \longrightarrow_1)$ and $(\overrightarrow{\mathsf{obs}}_2, \longrightarrow_2)$, a relation $R \subseteq X \times Y$ is weak simulation if the following is true for all $x \in X, y \in Y$ with $R(x, y)$:

$$\mathsf{obs}_1(x) = \mathsf{obs}_2(y) \wedge (\forall x.\, x \longrightarrow_1 x' \implies \exists y'.y \longrightarrow_2^* y' \wedge R(x', y')).$$

This says that two states x, y are related if they produce the same observables and the system $\longrightarrow_2$ can "match" the transitions of x, in potentially many steps.

Notation 26. Our coalgebras are transition systems where the state-space has been augmented with events, e.g., $\langle \alpha, x\rangle \in \mathcal{E}_1 \times X$ and $\langle \beta, y\rangle \in \mathcal{E}_2 \times Y$. We say a pair $(\approx, Q) \subseteq (\mathcal{E}_1 \times \mathcal{E}_2) \times (X \times Y)$ is a (weak) simulation if the relation R defined by $(\langle \alpha, x\rangle, \langle \beta, y\rangle) \in R \iff \alpha \approx \beta \wedge (x, y) \in Q$ is a (weak) simulation.

4.2 Emulation of state-based CRDTs by op-based CRDTs

Shapiro et al. provides a simple way to turn a state-based CRDT to an op-based CRDT. Given given any state-based CRDT

$$c = ((S, \sqcup), s_0, \mathtt{u}, \mathtt{q}), \tag{8}$$

we can construct the op-based CRDT $\mathcal{F}(c)$ by

$$c \overset{\mathcal{F}}{\longmapsto} (S, s_0, S, \mathtt{u}, \mathtt{u}, \sqcup, \mathtt{q}). \tag{9}$$

Indeed, the tuple $(S, s_0, S, \mathtt{u}, \mathtt{u}, \mathtt{q})$ is an instance of Definition 2. Applied to CRDTs c and $\mathcal{F}(c)$, the semantics introduced in Section 3.1 and Section 3.2 yield resp. the CRDT systems $(\rightsquigarrow_{\mathsf{op}}, q)$ and $(\rightsquigarrow_{\mathsf{st}}, q)$ which, according to Remarks 17 and 20, induce the coalgebras

$$(\rightsquigarrow_{\mathsf{op}}, \vec{q}) : \mathcal{E}_{\mathsf{op}} \times (S \times \mathcal{M}(S))^n \to \mathcal{P}(\mathcal{E}_{\mathsf{op}} \times (S \times \mathcal{M}(S))^n) \times B^n$$
$$(\rightsquigarrow_{\mathsf{st}}, \vec{q}) : \mathcal{E}_{\mathsf{st}} \times (S \times \mathcal{M}(S))^n \to \mathcal{P}(\mathcal{E}_{\mathsf{st}} \times (S \times \mathcal{M}(S))^n) \times B^n.$$

We now state the equivalence between CRDT systems c and $\mathcal{F}(c)$ in terms of two weak simulations: one of $(\rightsquigarrow_{\mathsf{op}}, \vec{q})\rangle$ by $(\rightsquigarrow_{\mathsf{st}}, \vec{q})$ and another one in the opposite direction.

First, construct the intermediate system as in Definition 22, whose transition arrows are denoted with $\longmapsto$. The next theorem and its subsequent corollary are essentially direct, so we omit their proofs.

Theorem 27. *Let $\Delta \subseteq \mathcal{E}_{\mathsf{op}} \times (S \times \mathcal{M}(S))^n \times \mathcal{E}_{\mathsf{op}} \times (S \times \mathcal{M}(S))^n$ be the diagonal relation, i.e.,*

$$\Delta = \{\langle \alpha, (x_i)_{i \in n}\rangle, \langle \alpha, (x_i)_{i \in n}\rangle \mid \alpha \in \mathcal{E}_{\mathsf{op}} \wedge (x_i)_{i \in n} \in S \times \mathcal{M}(S))^n\}.$$

Relation Δ is a simulation of $(\rightsquigarrow_{\mathsf{op}}, \vec{q})$ and $(\longmapsto, \vec{q})$.

Corollary 28. *There is a weak simulation $\mathcal{Q}_1$ of $(\rightsquigarrow_{\mathsf{op}}, \vec{q})$ and $(\rightsquigarrow_{\mathsf{st}}, \vec{q})$ arising from Δ which contains the initial states $\langle \top, (s^0, \varnothing)_{i\in n}\rangle \in \mathcal{E}_{\mathsf{op}} \times (S \times \mathcal{M}(S))^n$ and $\langle \top, (s^0, \varnothing)_{i\in n}\rangle \in \mathcal{E}_{\mathsf{st}} \times (S \times \mathcal{M}(S))^n$.*

$\mathcal{Q}_1$ is a weak simulation of $(\rightsquigarrow_{\mathsf{op}}, \vec{q})$ and $(\rightsquigarrow_{\mathsf{st}}, \vec{q})$, but it is not a weak simulation of $(\rightsquigarrow_{\mathsf{st}}, \vec{q})$ and $(\rightsquigarrow_{\mathsf{op}}, \vec{q})$, as it is too strict a relation for this purpose. There is, however, a larger relation that is so. To define it, we use the following auxiliary definition.

Definition 29. Let $\mathcal{F}(c) = (S, s_0, S, \mathtt{u}, \mathtt{u}, \sqcup, \mathtt{q})$ be the aforementioned op-based emulator. We say a state $(s_i, \tau_i)_{i\in n}$ of $\mathcal{F}(c)$ is *synchronizable* if

$$\forall i, j \in n, C \subseteq \tau_i. \exists D \in \tau_j. \bigsqcup(\{s_i\} \cup \{s_j\} \cup C) = \bigsqcup(\{s_j\} \cup D). \tag{10}$$

Roughly, a state being synchronizable means that any replica can "catch up" with any other replica by delivering one or more messages from its buffer. The next proposition foreshadows how being synchronizable plays a role during a weak simulation.

Proposition 30. *Let $(x_i)_{i\in n}$ be a synchronizable state of the op-based CRDT system $(\rightsquigarrow_{\mathsf{op}}, \mathtt{q})$ induced by $\mathcal{F}(c)$.*

The following is true: if $\langle \alpha, (x_i)_{i\in n}\rangle \rightsquigarrow^*_{\mathsf{op}} \langle \alpha', (x'_i)_{i\in n}\rangle$, and $\alpha' = \mathtt{bc}^j(m)$ or $\alpha' = \mathtt{dlvr}^{j\leftarrow k}(m)$, then $(x'_i)_{i\in n}$ is synchronizable. Otherwise $\alpha' = \mathtt{upd}^j(a)$, and

$$\langle \mathtt{upd}^j(a), (x'_i)_{i\in n}\rangle \rightsquigarrow_{\mathsf{op}} \langle \mathtt{bc}^j(m), z\rangle$$

where $\langle \mathtt{bc}^j(m), z\rangle$ is synchronizable.

We are now ready to construct our weak simulation relation. First, we define a relation $\mathcal{Q}_2 \in (S \times \mathcal{M}(S))^n \times (S \times \mathcal{M}(S))^n$ on the state spaces of $(\rightsquigarrow_{\mathsf{st}}, \vec{q})$ and $(\rightsquigarrow_{\mathsf{op}}, \vec{q})$ as follows:

$$\begin{aligned} \mathcal{Q}_2 = \big\{&((s_i, \sigma_i)_{i\in n}, (s_i, \tau_i)_{i\in n}) \mid \\ &\forall i, j \in n. (\forall s \in \sigma_i. \exists B \subseteq \tau_i. s_i \sqcup s = \bigsqcup(\{s_i\} \cup B)) \\ &\wedge (s_i, \tau_i)_{i\in n} \text{ is synchronizable}\big\}. \end{aligned} \tag{11}$$

The intuition behind the $\mathcal{Q}_2$ relation is that when $(x, y) \in \mathcal{Q}_2$, the coalgebraic state y is "ahead" of x in terms of synchronization and, furthermore, it is always in the optimal synchronization state where all internal replica states converge, at the same internal state, after delivering all messages. Next, we let $\approx \subseteq \mathcal{E}_{\mathsf{st}} \times \mathcal{E}_{\mathsf{op}}$ be a relation on events, defined as follows:

(i). $\mathtt{upd}^j(a) \approx \mathtt{bc}^j(m)$ for $a \in A$, $m \in M$,
(ii). $\mathtt{dlvr}^{j\leftarrow i}(s) \approx \mathtt{dlvr}^{j\leftarrow i}(s')$ for $s, s' \in S$,
(iii). $\mathtt{send}^{j\rightarrow i}(s) \approx \alpha$, for all $\alpha \in \mathcal{E}_{\mathsf{op}}$.

Theorem 31. *Relation $(\approx, \mathcal{Q}_2)$ is a weak simulation of $(\rightsquigarrow_{\mathsf{st}}, \vec{q})$ and $(\rightsquigarrow_{\mathsf{op}}, \vec{q})$.*

Proof sketch. Let $(x_i)_{i\in n} = (s_i, \sigma_i)_{i\in n}$ and $(y_i)_{i\in n} = (s_i, \tau_i)_{i\in n}$ such that for a pair of events $\beta \in \mathcal{E}_{\mathsf{st}}, \alpha \in \mathcal{E}_{\mathsf{op}}$, $((x_i)_{i\in n}, (y_i)_{i\in n}) \in \mathcal{Q}_2$ and $\beta \approx \alpha$. By Definitions 23 and 24, we have to show

$$\vec{q}((x_i)_{i\in n}) = \vec{q}((y_i)_{i\in n}) \tag{12}$$

$$\begin{aligned} &\langle \beta, (x_i)_{i\in n}\rangle \rightsquigarrow_{\mathsf{st}} \langle \beta', w\rangle \\ \implies &\exists \alpha', z.\, \langle \alpha, (y_i)_{i\in n}\rangle \rightsquigarrow^*_{\mathsf{op}} \langle \alpha', z\rangle \wedge\ \alpha' \approx \beta' \wedge (w, z) \in \mathcal{Q}_2. \end{aligned} \tag{13}$$

Equation (12) is immediate, as $\vec{q}$ only depends on the internal states s_i, which are the same in both replica states x_i and y_i for all i. For (13), we proceed by case distinction on the global semantics of state-based systems (7). We identify and sketch three cases on the transition $\langle \beta, (x_i)_{i\in n}\rangle \rightsquigarrow_{\mathsf{st}} \langle \beta', w\rangle$.

1. ($\beta' = \mathtt{upd}^j(a)$). There are two sub-cases: if we have the sub-case $\alpha \notin \mathtt{update}$, then simulate with the transitions

   $$\langle \alpha, (y_i)_{i\in n}\rangle \rightsquigarrow_{\mathsf{op}} \langle \mathtt{upd}^j(a), (s'_i, \tau'_i)_{i\in n}\rangle \rightsquigarrow_{\mathsf{op}} \langle \mathtt{bc}^j(s'_j), z\rangle,$$

   where $s'_j = \mathtt{u}(s_j, a)$ in the previous step. Then $\mathtt{upd}^j(a) \approx \mathtt{bc}^j(m)$ and by Proposition 30, z is synchronizable. We thus have to show $\forall i \in n, i \neq j$,

   $$\forall s \in \sigma_i.\, \exists B \subseteq \tau_i \cup \{s_j\}.\, s_i \sqcup s = \bigsqcup(\{s_i\} \cup B). \tag{14}$$

   $$\forall s \in \sigma_j.\, \exists B \subseteq \tau_j.\, \mathtt{u}(s_j, a) \sqcup s = \bigsqcup(\{\mathtt{u}(s_j, a)\} \cup B). \tag{15}$$

   Statement (14) immediately holds because of $((x_i)_{i\in n}, (y_i)_{i\in n}) \in \mathcal{Q}_2$. Statement (15) follows from the fact that $\forall s \in S,\ s \sqcup \mathtt{u}(s, a) = \mathtt{u}(s, a)$ in conjunction with $((x_i)_{i\in n}, (y_i)_{i\in n}) \in \mathcal{Q}_2$. This closes the sub-case. For the other sub-case that $\alpha = \mathtt{upd}^k(a')$, we must first complete the update-broadcast cycle (recall they are essentially atomic in the semantics 4), then perform the simulating transitions above, and apply the same argument. This close the second sub-case.
2. ($\beta' = \mathtt{send}^{j\to i}(s)$). If $\alpha \in \mathtt{update}$, then finish the update-broadcast cycle yielding configuration $\langle \alpha', z\rangle$ where α' is a broadcast event, and z is synchronizable by Proposition 30. Then simulate with the reflexive step $\langle \alpha', z\rangle \rightsquigarrow^*_{\mathsf{op}} \langle \alpha', z\rangle$. The remaining details follow similarly to the previous case. Note that if $\alpha \notin \mathtt{update}$ then the reflexive step alone suffices.
3. ($\beta' = \mathtt{dlvr}^{j\leftarrow i}(s)$). We need to simulate by delivering the right sequence of messages. It thus suffices to only deal with the $\alpha \notin \mathtt{update}$ sub-case, since finishing the update-broadcast cycle would yield buffers τ'_i which would contain the buffers τ_i as subsets had we not done this. Hence the proof for $\alpha \notin \mathtt{update}$ implies the other subcase $\alpha \in \mathtt{update}$.
   The event $\mathtt{dlvr}^{j\leftarrow i}(s)$ implies $s \in \sigma_j$, so from the hypothesis, there is a subset $B \subseteq \tau_j$ so that $s_j \sqcup s = \bigsqcup(\{s_j\} \cup B)$. Therefore, there is a (possibly empty) trace (Definition 15) $\langle \alpha_i, z^i\rangle_{i\in k}$ from $\langle \alpha, (y_i)_{i\in n}\rangle$ s.t. the $(\alpha_i)_{i\in k}$ are all deliver events on replica j, and thus correspond to the subset $B \subseteq \tau_j$. One can show that the resulting global state z^k is synchronizable, and replica j in z^k agrees with replica j in w. The simulating transitions is thus implied by the trace $\langle \alpha_i, z^i\rangle_{i\in k}$. ◻

4.3 Emulation of op-based CRDTs by state-based CRDTs

The next translation is from op-based to state-based CRDTs, which is a slightly modified version of the one in Shapiro et al. [43]. The original translation takes a given op-based CRDT $o = (S, s_0, M, \mathtt{u_{op}}, \mathtt{t}, \mathtt{e}, \mathtt{q_{op}})$ and translates it into a state-based CRDT with replica state space $S \times \mathcal{P}_{fin}(M) \times \mathcal{P}_{fin}(M)$. The emulating replica state is thus a triple (s, H, D) of an internal state, a set of known messages H, and a set of delivered messages D. Merges on the emulating system work by creating a new set of known messages on the merged replica via set-theoretic union. Finally, updating is implemented in terms of some recursive function d which, apart from applying a given operation, also delivers qualified messages that are still in H.

In our modified translation, we observe that it is sufficient to simply use the sets $H \in \mathcal{P}_{fin}(M)$ as the emulating states, since one can recover the states s on-demand by computation (thus also eliminating the need for D). The idea is that, since M is equipped with the causal relation $\prec$ (a partial order - see Remark 16), H essentially represents an equivalence class of strings of messages $m_1 m_2 \cdots m_{|H|}$, which all result in the same end state s when applied to the initial state s^0.

Op-based to state-based CRDT translation. We show how the aforementioned H sets arise in our semantics.

Let $(S, s_0, M, \mathtt{u_{op}}, \mathtt{t}, \mathtt{e}, \mathtt{q_{op}})$ be an op-based CRDT, and $(\rightsquigarrow_{\mathtt{op}}, \mathtt{q_{op}})$ its corresponding system, taking $\langle \top, (s^0, \varnothing)_{i \in n} \rangle$ as the initial configuration. Following Remark 16, any trace $\langle \alpha_i, z^i \rangle_{i \in k}$ of $\langle \top, (s^0, \varnothing)_{i \in n} \rangle$ (Definition 15) yields a partially ordered (by causality) set E of events with partition $(E_i)_{i \in n}$.

Then there is an obvious map $E_i \mapsto H_i \in \mathcal{P}_{fin}(M)$ which projects the event sets down to finite message sets, inheriting a partial order $\prec$ (we assume unique events project to unique messages), and thus also the concurrency relation $\parallel$. That is, $m \parallel m' \iff (m \not\prec m') \wedge (m' \not\prec m)$. We call H_i the *history* for replica i, and they can be thought of as denoting states $s_i \in S$ by the following lemma.

Lemma 32. *Let $(M, \prec)$ be the partial order given by causality, and let $s^0 \in S$ be a local state. Then, there is a well-defined map $[\![\cdot]\!] : \mathcal{P}_{fin}(M) \to S$.*

Proof sketch. Define the equivalence relation $w \equiv_{\mathtt{e}} w' \iff \mathtt{e}_w = \mathtt{e}_{w'}$ (Notation 3). By Remark 4 and Remark 16, for all $m, m' \in M$ if $m \parallel m'$ then necessarily $m \cdot m' \equiv_{\mathtt{e}} m' \cdot m$. We note the following:

1. There is a map $H \mapsto \mathcal{L}(H)$, where $\mathcal{L}(H) \in \mathcal{P}_{fin}(M^*)$ is the set of *linear extensions* of $\prec$ on H s.t. $\forall w, w' \in \mathcal{L}(H)$, we have $w \equiv_{\mathtt{e}} w'$.
2. Hence, there is a map $\mu : \mathcal{P}_{fin}(M) \to M^* / \equiv_{\mathtt{e}}$ which sends $H \overset{\mu}{\mapsto} [w]_{\equiv_{\mathtt{e}}}$, and an injection $g : M^* / \equiv_{\mathtt{e}} \to M$ which selects a representative $w' \in [w]_{\equiv_{\mathtt{e}}}$.
3. Finally, we define $[\![H]\!] = \mathtt{e}_{w'}(s^0)$ where $w' = (g \circ \mu)(H)$.

The map $[\![\cdot]\!]$ is well-defined in the sense that it does not depend on choice of $w' \in \mu(H)$. $\square$

Corollary 33. *Let $[\![\cdot]\!] : \mathcal{P}_{fin}(M) \to S$ be given by Lemma 32, and let $H \in \mathcal{P}_{fin}(M)$ be a partially ordered set. For all $m \in M$,*

$$(\forall m' \in H.\, (m' \prec m) \vee (m' \parallel m)) \implies [\![H \cup \{m\}]\!] = \mathsf{e}([\![H]\!], m).$$

Proof sketch. If w is any linear extension of $(H, \prec)$, and if $m' \prec m$ or $m' \parallel m$ holds $\forall m' \in H$, then any linear extension of w' of $H \cup \{m\}$ belongs to the equivalence class $[w \cdot m]_{\equiv_{\mathsf{e}}}$, hence $\mathsf{e}_{w'}(s^0) = \mathsf{e}_{w \cdot m}(s^0) = \mathsf{e}([\![H]\!], m)$. ☐

In all that follows, we assume for simplicity fixed message set $(M, \prec)$ and initital state s^0. We can apply Lemma 32 to $(M, \prec)$ and s^0 to obtain an *interpretation function* $[\![\cdot]\!] : \mathcal{P}_{fin}(M) \to S$ and construct the state-based CRDT $\mathcal{G}(o)$ as follows:

$$o \overset{\mathcal{G}}{\mapsto} ((\mathcal{P}_{fin}(M), \cup), \varnothing, \mathsf{u}_{\mathsf{st}}, \mathsf{q}_{\mathsf{st}}), \tag{16}$$

where $\mathsf{u}_{\mathsf{st}}(H, a) = H \cup \{\mathsf{t}([\![H]\!], a)\}$ and $\mathsf{q}_{\mathsf{st}}(H) = \mathsf{q}([\![H]\!])$. The fact that $\mathcal{G}(o)$ emulates o is a straightforward corollary of Lemma 32. In particular, definition (16) induces the CRDT systems $(\rightsquigarrow_{\mathsf{op}}, \mathsf{q}_{\mathsf{op}})$ and $(\rightsquigarrow_{\mathsf{st}}, \mathsf{q}_{\mathsf{st}})$, which weakly simulate one another.

The weak simulation argument consists of several steps. First, we construct an intermediate op-based CRDT system $(\longrightarrow, \mathsf{q})$ which behaves the same as the original system $(\rightsquigarrow_{\mathsf{op}}, \mathsf{q}_{\mathsf{op}})$, except now each replica state additionally carries not only state s_i, but also the partially ordered (by causality) set of d_i of *delivered* messages such that $[\![d_i]\!] = s_i$ (Lemma 32) holds as an *invariant*. Second, we also pass from $(\rightsquigarrow_{\mathsf{st}}, \mathsf{q}_{\mathsf{st}})$ to an intermediate system $(\longmapsto, \mathsf{q}')$ via Definition 22, also with histories H_i as the replica states.

We then show a weak simulation from $(\longrightarrow, \mathsf{q})$ to $(\longmapsto, \mathsf{q}')$ which maintains the equality $d_i = H_i$, hence $[\![H_i]\!] = s_i$.

We now construct the intermediate $(\longrightarrow, \mathsf{q})$ for the $(\rightsquigarrow_{\mathsf{op}}, \mathsf{q}_{\mathsf{op}})$ system. Define the map

$$o \mapsto (S \times \mathcal{P}_{fin}(M), (s_0, \varnothing), M, \mathsf{u}, \mathsf{t}', \mathsf{e}', \mathsf{q}) \tag{17}$$

where $\mathsf{t}'((s,d), a) = \mathsf{t}(s, a)$, and $\mathsf{e}'((s,d), m) = (\mathsf{e}(s, m), d \cup \{m\})$, and $\mathsf{u}(z, a) = \mathsf{e}'(z, \mathsf{t}'(z, a))$. Finally, set $\mathsf{q}(s, d) = \mathsf{q}_{\mathsf{op}}(s)$. This induces an intermediate op-based system $(\longrightarrow, \mathsf{q})$ with the same set of events $\mathcal{E}_{\mathsf{op}}$ as $(\rightsquigarrow_{\mathsf{op}}, \mathsf{q}_{\mathsf{op}})$ system induced from o.

The next lemma expresses that $(\longrightarrow, \mathsf{q})$ has the same behavior as $(\rightsquigarrow_{\mathsf{op}}, \mathsf{q}_{\mathsf{op}})$.

Lemma 34. *The forgetful map*

$$f : \mathcal{E}_{\mathsf{op}} \times (S \times \mathcal{P}_{fin}(M) \times \mathcal{M}(M))^n \to \mathcal{E}_{\mathsf{op}} \times (S \times \mathcal{M}(M))^n$$
$$f(\alpha, (s_i, d_i, \sigma_i)_{i \in n}) = \langle \alpha, (s_i, \sigma_i)_{i \in n} \rangle$$

is a coalgebra homomorphism, hence its graph a bisimulation.

Lemma 32 and Corollary 33 establishes the relation between the s_i states and the delivered messages d_i, summarized as a lemma.

Lemma 35. *Define* $\mathsf{Reach}(\alpha, z) = \{\langle \alpha', z' \rangle \mid \langle \alpha, z \rangle \longrightarrow^* \langle \alpha', z' \rangle\}$ *as the set of reachable states from* $\langle \alpha, z \rangle$*. Then the predicate* $P \subseteq \mathcal{E}_{\mathsf{op}} \times (S \times \mathcal{P}_{fin}(M) \times \mathcal{M}(M))^n$ *defined by*

$$P = \{\langle \alpha, (s_i, d_i, \sigma_i)_{i \in n} \rangle \mid \forall i \in n.\ s_i = [\![d_i]\!]\} \cap \mathsf{Reach}(\top, (s^0, \varnothing, \varnothing)) \tag{18}$$

is non-empty and an invariant of the intermediate system $(\longrightarrow, \mathtt{q})$*, where* $[\![\cdot]\!] : \mathcal{P}_{fin}(M) \to S$ *is given by Lemma 32.*

Lemmas 34 and 35 allow us to denote a replica state s_i by a set of delivered messages d_i under the op-based semantics. This informs us that the desired simulation is that the state-based semantics can equivalently represent the delivered set d_i, though perhaps with multiple steps, and ignoring event labels.

To that end, apply Definition 22 and pass from $(\rightsquigarrow_{\mathsf{st}}, \mathsf{q}_{\mathsf{st}})$ to the intermediate system $(\longmapsto, \mathtt{q}')$ (so that our events $\mathcal{E}$ correspond to the broadcasting and delivering of *histories* of messages).

Our weak simulation argument is dependent on the relation $\approx\, \subseteq \mathcal{E}_{\mathsf{op}} \times \mathcal{E}$ on events, which we promptly define. Let $m \in M$ and define $H(m) \in \mathcal{P}_{fin}(M)$ as the finite downward closed set of messages wrt $\prec$ with m at the top.[4] We inductively define $\approx\, \subseteq \mathcal{E}_{\mathsf{op}} \times \mathcal{E}$ as follows:

$$\begin{aligned} \mathtt{upd}^i(a) &\approx \mathtt{upd}^i(a) \\ \mathtt{bc}^i(m) &\approx \mathtt{bc}^i(H(m)) \\ \mathtt{dlvr}^{j \leftarrow i}(m) &\approx \mathtt{dlvr}^{j \leftarrow i}(H(m)). \end{aligned} \tag{19}$$

We now give the statement of the main theorem and sketch its proof.

Theorem 36. *Let* $\mathcal{H} = \mathcal{P}_{fin}(M)$*. For all states* s_i*, write* $(s_i, d_i) \sim_1 H_i \iff [\![H_i]\!] = s_i = [\![d_i]\!]$*. Also write* $\sigma_i \sim_2 \tau_i$ *if* $m \in \sigma_i \iff H(m) \in \tau_i$*. Then the pair of relations*

$$(\approx, \mathcal{R}_1) \subseteq (\mathcal{E}_{\mathsf{st}} \times \mathcal{E}) \times ((\mathcal{H} \times \mathcal{M}(\mathcal{H}))^n \times (S \times \mathcal{H} \times \mathcal{M}(M))^n)$$

where $\mathcal{R}_1 = \prod_{i \in n} R_i$*, and*

$$R_i = \{(s_i, d_i, \sigma_i), (H_i, \tau_i) \mid (s_i, d_i) \sim_1 H_i \wedge \sigma_i \sim_2 \tau_i\}, \tag{20}$$

is a simulation of $(\longrightarrow, \mathtt{q})$ *and* $(\longmapsto, \mathtt{q}')$*.*

Proof sketch. On related configurations $\langle \alpha, (s_i, d_i, \sigma_i)_{i \in n} \rangle$ and $\langle \beta, (H_i, \tau_i)_{i \in n} \rangle$, proceed by case analysis on the transitions $\langle \alpha, (s_i, d_i, \sigma_i) \rangle \longrightarrow \langle \gamma, (y_i)_{i \in n} \rangle$ and find the simulating transition $\langle \beta, (H_i, \tau_i)_{i \in n} \rangle \longmapsto \langle \phi, (z_i)_{i \in n} \rangle$. The cases are:

1. $\gamma = \mathtt{upd}^j(a)$, then y_j has updated both s_j and d_j by generating locally and applying the message m. This same message m can be generated and applied locally from (H_j, τ_j) and event-transition $\phi = \mathtt{upd}^j(a)$.

[4] We may essentially think of $H(m)$ as a *causal history* [42] of m.

2. $\gamma = \texttt{bc}^j(m)$, then each $y_i \neq y_j$ has input m in their buffer. From the op-based rules, we must have $\alpha = \texttt{upd}^j(a)$ as well and therefore $\beta = \texttt{upd}^j(a)$. By the previous case, and the semantics of causality, $H_j = H(m)$, so we choose $\phi = \texttt{bc}^j(H_j)$.
3. $\gamma = \texttt{dlvr}^{j \leftarrow k}(m)$, then $m \in \sigma_j$ and was $\mathsf{deliverable}$. Apply the hypothesis and do the corresponding $\phi = \texttt{dlvr}^{j \leftarrow k}(H(m))$ event. □

The following corollary coincides with weak simulation, and is ultimately what we wanted to show.

Corollary 37. *For every execution* $\langle \top, (s^0, \varnothing)_{i \in n} \rangle \rightsquigarrow^*_{\mathsf{op}} \langle \alpha, (s_i, \sigma_i)_{i \in n} \rangle$*, there is a corresponding execution* $\langle \top, (\varnothing, \varnothing)_{i \in n} \rangle \rightsquigarrow^*_{\mathsf{st}} \langle \beta, (H_i, \tau_i)_{i \in n} \rangle$ *where* $\forall i \in n.\, [\![H_i]\!] = s_i$*. Hence every* $(\rightsquigarrow_{\mathsf{op}}, \mathsf{q}_{\mathsf{op}})$ *behavior yields a* $(\rightsquigarrow_{\mathsf{st}}, \mathsf{q}_{\mathsf{st}})$ *behavior.*

The other weak simulation can be argued more directly. It's proof is remarkably similar to that of Theorem 31. Define the relation $\approx$ on events $\mathcal{E}_{\mathsf{st}} \times \mathcal{E}_{\mathsf{op}}$ as follows:

(i). $\texttt{upd}^j(a) \approx \texttt{bc}^j(m)$ for $a \in A$, $m \in M$,
(ii). $\texttt{dlvr}^{j \leftarrow i}(H) \approx \texttt{dlvr}^{j \leftarrow i}(m)$ for $H \in \mathcal{P}_{fin}(M)$ and $m \in M$,
(iii). $\texttt{send}^{j \rightarrow i}(H) \approx \alpha$, for all $\alpha \in \mathcal{E}_{\mathsf{op}}$.

Theorem 38. *Define the relation* $\mathcal{R}_2$ *on global states as*

$$\mathcal{R}_2 = \prod\nolimits_{i \in n} \{(H_i, \tau_i), (s_i, d_i, \sigma_i) \mid (s_i, d_i) \sim_1 H_i \wedge \tau_i \sim_2 \sigma_i\}$$

where $\sim_1$ *is as in Theorem 36 and*

$$\sigma_i \sim_2 \tau_i \iff \forall H \in \tau_i.\, \exists \{m_1, \ldots, m_k\} \subseteq \sigma_i.\, [\![H_i \cup H]\!] = \mathsf{e}_{m_1 \cdots m_k}(s_i).$$

Then $(\approx, \mathcal{R}_2)$ *is a weak simulation from* $(\rightsquigarrow_{\mathsf{st}}, \mathsf{q}_{\mathsf{st}})$ *to* $(\rightsquigarrow_{\mathsf{op}}, \mathsf{q}_{\mathsf{op}})$*.*

Proof sketch. Follow the proof structure of Theorem 31. The only tricky case is when we must deliver a sequence of messages $m_1 \cdots m_k$ to simulate the delivery of a set $H \in \mathcal{P}_{fin}(M)$. The key insight is that H itself contains only a subset of messages $O \subseteq H$ which produce an observable effect in the computation. That is, $[\![H_i \cup H]\!] = [\![H \cup O]\!]$. The set O of messages is the set $\{m_1, \ldots, m_k\}$ we are looking for. □

5 Related Work

Specification and verification of CRDTs. The primary focus of the most existing research on verified CRDTs has been the verification of strong convergence and other safety properties for either state-based [45, 12, 31, 44, 33, 22] or op-based [15, 30, 25, 24, 32] CRDTs.[5] One exception is the work of Burckhardt et

[5] Timany et al. [44] also consider verification of *liveness* properties, such as eventual delivery of messages.

al. [9], who give a framework for axiomatic specification and verification of both op-based and state-based CRDTs, inspired by previous work on axiomatization of weak memory models.

To our knowledge, our work is the first to take the approach of modeling CRDTs with coalgebra in mind. However, the use of (bi)simulation relations in CRDT verification is not new. For instance, Burckhardt et al. [9]'s framework is based on *replication-aware simulations*, and Nair et al. [31] use a strong bisimulation argument to justify the use of simpler, easier-to-implement proof rules in an automated verification tool for state-based CRDTs, using the more complicated semantics as a reference implementation.

Nieto et al. [33] observe that whether a CRDT is op-based or state-based is in fact an implementation detail that should be hidden from clients, and their verification approach centers around this *representation independence* property. They contribute the first mechanically verified representation independence result for a specific CRDT, the *pn-counter*, demonstrating that a particular client program cannot distinguish between (handwritten) op-based and state-based implementations of this CRDT. Our goal in this work, on the other hand, is to make precise the sense in which Shapiro et al. [43]'s *general* op-to-state-based and state-to-op-based emulation algorithms result in the same observable behavior for the original and the emulating object. However, we have not yet attempted any proof mechanization. More generally, mechanized verification of state-based and op-based CRDTs, both interactive [45, 15, 44, 32, 33] and automated [31, 30, 10, 22], is an active area of research. Our goal in this work is to complement these existing verification efforts by making precise the sense in which results for op-based CRDTs can be said to transfer to state-based CRDTs and vice versa.

Coalgebraic reasoning about concurrent and distributed systems. The theory of universal coalgebra [20, 37, 36, 16] is a general model for state-based systems, and can be seen as a generalization of classical work on process calculi [38, 17, 28, 29]. The typical notion of equivalence when modeling a system as a process calculus or as a coalgebra is that of bisimulation [38, 28, 29], since classical definitions of bisimulation on process calculi generalize nicely to coalgebras as Hermida-Jacobs bisimulation [20], or Aczel-Mendler bisimulations [2, 37].

The transition system coalgebras we give here are presented on a mostly semantic level, i.e., there is no corresponding process calculus. Thus our models of CRDTs are closer in spirit to the the classical state-machine models of distributed computing theory [23, 41, 4, 26]. Moreover, the concept of simulation (as opposed to bisimulation) in coalgebra and classical distributed computing theory is not new [19, 4, 1], although, unlike in the present work, classical distributed computing theory does not make use of the aforementioned coalgebraic notion.

References

[1] Martin Abadi and Leslie Lamport. “The Existence of Refinement Mappings”. In: *Proceedings of the 3rd Annual Symposium on Logic in Computer Science*. LICS 1988 Test of Time Award. 1988, pp. 165–175. URL: https://www.microsoft.com/en-us/research/publication/the-existence-of-refinement-mappings/.

[2] Peter Aczel and Nax Mendler. “A final coalgebra theorem”. In: *Category Theory and Computer Science*. Ed. by David H. Pitt et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 1989, pp. 357–365. ISBN: 978-3-540-46740-3.

[3] Paulo Sérgio Almeida, Ali Shoker, and Carlos Baquero. “Efficient State-Based CRDTs by Delta-Mutation”. In: *Networked Systems*. Ed. by Ahmed Bouajjani and Hugues Fauconnier. Cham: Springer International Publishing, 2015, pp. 62–76. ISBN: 978-3-319-26850-7.

[4] Hagit Attiya and Jennifer Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2004. ISBN: 0471453242.

[5] Falk Bartels, Ana Sokolova, and Erik P. de Vink. “A hierarchy of probabilistic system types”. In: *6th International Workshop on Coalgebraic Methods in Computer Science, CMCS 2003, Satellite Event for ETAPS 2003, Warsaw, Poland, April 5-6, 2003*. Ed. by H. Peter Gumm. Vol. 82. Electronic Notes in Theoretical Computer Science 1. Elsevier, 2003, pp. 57–75. DOI: 10.1016/S1571-0661(04)80632-7. URL: https://doi.org/10.1016/S1571-0661(04)80632-7.

[6] K. Birman and T. Joseph. “Exploiting Virtual Synchrony in Distributed Systems”. In: *SIGOPS Oper. Syst. Rev.* 21.5 (Nov. 1987), pp. 123–138. ISSN: 0163-5980. DOI: 10.1145/37499.37515. URL: https://doi.org/10.1145/37499.37515.

[7] Kenneth Birman, André Schiper, and Pat Stephenson. “Lightweight Causal and Atomic Group Multicast”. In: *ACM Trans. Comput. Syst.* 9.3 (Aug. 1991), pp. 272–314. ISSN: 0734-2071. DOI: 10.1145/128738.128742. URL: https://doi.org/10.1145/128738.128742.

[8] Kenneth P. Birman and Thomas A. Joseph. “Reliable Communication in the Presence of Failures”. In: *ACM Trans. Comput. Syst.* 5.1 (Jan. 1987), pp. 47–76. ISSN: 0734-2071. DOI: 10.1145/7351.7478. URL: https://doi.org/10.1145/7351.7478.

[9] Sebastian Burckhardt et al. “Replicated Data Types: Specification, Verification, Optimality”. In: *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. POPL ’14. San Diego, California, USA: Association for Computing Machinery, 2014, pp. 271–284. ISBN: 9781450325448. DOI: 10.1145/2535838.2535848. URL: https://doi.org/10.1145/2535838.2535848.

[10] Kevin De Porre, Carla Ferreira, and Elisa Gonzalez Boix. “VeriFx: Correct Replicated Data Types for the Masses”. In: *37th European Conference on Object-Oriented Programming (ECOOP 2023)*. Ed. by Karim Ali and

Guido Salvaneschi. Vol. 263. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, 9:1–9:45. ISBN: 978-3-95977-281-5. DOI: 10.4230/LIPIcs.ECOOP.2023.9. URL: https://drops.dagstuhl.de/opus/volltexte/2023/18202.

[11] C. J. Fidge. "Timestamps in message-passing systems that preserve the partial ordering". In: *Proceedings of the 11th Australian Computer Science Conference* 10.1 (1988), pp. 56–66.

[12] Fabio Gadducci, Hernán Melgratti, and Christian Roldán. "On the semantics and implementation of replicated data types". In: *Science of Computer Programming* 167 (2018), pp. 91–113. ISSN: 0167-6423. DOI: https://doi.org/10.1016/j.scico.2018.06.003. URL: https://www.sciencedirect.com/science/article/pii/S0167642318302429.

[13] Seth Gilbert and Nancy Lynch. "Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services". In: *SIGACT News* 33.2 (2002), pp. 51–59. ISSN: 0163-5700. DOI: 10.1145/564585.564601. URL: https://doi.org/10.1145/564585.564601.

[14] Seth Gilbert and Nancy Lynch. "Perspectives on the CAP Theorem". In: *Computer* 45.2 (2012), pp. 30–36. DOI: 10.1109/MC.2011.389.

[15] Victor B. F. Gomes et al. "Verifying Strong Eventual Consistency in Distributed Systems". In: *Proc. ACM Program. Lang.* 1.OOPSLA (2017). DOI: 10.1145/3133933. URL: https://doi.org/10.1145/3133933.

[16] Sergey Goncharov et al. "Towards a Higher-Order Mathematical Operational Semantics". In: *Proc. ACM Program. Lang.* 7.POPL (2023). DOI: 10.1145/3571215. URL: https://doi.org/10.1145/3571215.

[17] Ichiro Hasuo, Bart Jacobs, and Ana Sokolova. "The Microcosm Principle and Concurrency in Coalgebra". In: *Foundations of Software Science and Computational Structures*. Ed. by Roberto Amadio. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 246–260. ISBN: 978-3-540-78499-9.

[18] Maurice P. Herlihy and Jeannette M. Wing. "Linearizability: A Correctness Condition for Concurrent Objects". In: *ACM Trans. Program. Lang. Syst.* 12.3 (1990), pp. 463–492. ISSN: 0164-0925. DOI: 10.1145/78969.78972. URL: https://doi.org/10.1145/78969.78972.

[19] Jesse Hughes and Bart Jacobs. "Simulations in coalgebra". In: *Theor. Comput. Sci.* 327.1-2 (2004), pp. 71–108. DOI: 10.1016/J.TCS.2004.07.022. URL: https://doi.org/10.1016/j.tcs.2004.07.022.

[20] Bart Jacobs. *Introduction to Coalgebra: Towards Mathematics of States and Observation*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2016.

[21] Bart Jacobs. *Introduction to Coalgebra: Towards Mathematics of States and Observation*. Vol. 59. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2016. ISBN: 9781316823187. DOI: 10.1017/CBO9781316823187. URL: https://doi.org/10.1017/CBO9781316823187.

[22] Shadaj Laddad et al. "Katara: Synthesizing CRDTs with Verified Lifting". In: *Proc. ACM Program. Lang.* 6.OOPSLA2 (2022). DOI: 10.1145/3563336. URL: https://doi.org/10.1145/3563336.

[23] Leslie Lamport. "Time, Clocks, and the Ordering of Events in a Distributed System". In: *Commun. ACM* 21.7 (July 1978), pp. 558–565. ISSN: 0001-0782. DOI: 10.1145/359545.359563. URL: http://doi.acm.org/10.1145/359545.359563.

[24] Hongjin Liang and Xinyu Feng. "Abstraction for Conflict-Free Replicated Data Types". In: *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*. PLDI 2021. Virtual, Canada: Association for Computing Machinery, 2021, pp. 636–650. ISBN: 9781450383912. DOI: 10.1145/3453483.3454067. URL: https://doi.org/10.1145/3453483.3454067.

[25] Yiyun Liu et al. "Verifying Replicated Data Types with Typeclass Refinements in Liquid Haskell". In: *Proc. ACM Program. Lang.* 4.OOPSLA (2020). DOI: 10.1145/3428284. URL: https://doi.org/10.1145/3428284.

[26] Nancy A Lynch and Mark R Tuttle. *An introduction to input/output automata*. 1988.

[27] Friedemann Mattern. "Virtual Time and Global States of Distributed Systems". In: *Parallel and Distributed Algorithms*. North-Holland, 1989, pp. 215–226.

[28] R. Milner. *A Calculus of Communicating Systems*. Berlin, Heidelberg: Springer-Verlag, 1982. ISBN: 0387102353.

[29] Robin Milner, Joachim Parrow, and David Walker. "A calculus of mobile processes, I". In: *Information and Computation* 100.1 (1992), pp. 1–40. ISSN: 0890-5401. DOI: https://doi.org/10.1016/0890-5401(92)90008-4. URL: https://www.sciencedirect.com/science/article/pii/0890540192900084.

[30] Kartik Nagar and Suresh Jagannathan. "Automated Parameterized Verification of CRDTs". In: *Computer Aided Verification*. Ed. by Isil Dillig and Serdar Tasiran. Cham: Springer International Publishing, 2019, pp. 459–477. ISBN: 978-3-030-25543-5.

[31] Sreeja S. Nair, Gustavo Petri, and Marc Shapiro. "Proving the Safety of Highly-Available Distributed Objects". In: *Programming Languages and Systems*. Ed. by Peter Müller. Cham: Springer International Publishing, 2020, pp. 544–571. ISBN: 978-3-030-44914-8.

[32] Abel Nieto et al. "Modular Verification of Op-Based CRDTs in Separation Logic". In: *Proc. ACM Program. Lang.* 6.OOPSLA2 (2022). DOI: 10.1145/3563351. URL: https://doi.org/10.1145/3563351.

[33] Abel Nieto et al. "Modular Verification of State-Based CRDTs in Separation Logic". In: *37th European Conference on Object-Oriented Programming (ECOOP 2023)*. Ed. by Karim Ali and Guido Salvaneschi. Vol. 263. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, 22:1–

22:27. ISBN: 978-3-95977-281-5. DOI: 10.4230/LIPIcs.ECOOP.2023.22. URL: https://drops.dagstuhl.de/opus/volltexte/2023/18215.

[34] Nuno Preguiça, Carlos Baquero, and Marc Shapiro. "Conflict-Free Replicated Data Types CRDTs". In: *Encyclopedia of Big Data Technologies*. Ed. by Sherif Sakr and Albert Zomaya. Cham: Springer International Publishing, 2018, pp. 1–10. ISBN: 978-3-319-63962-8. DOI: 10.1007/978-3-319-63962-8_185-1. URL: https://doi.org/10.1007/978-3-319-63962-8_185-1.

[35] Hyun-Gul Roh et al. "Replicated abstract data types: Building blocks for collaborative applications". In: *Journal of Parallel and Distributed Computing* 71.3 (2011), pp. 354–368. ISSN: 0743-7315. DOI: https://doi.org/10.1016/j.jpdc.2010.12.006. URL: https://www.sciencedirect.com/science/article/pii/S0743731510002716.

[36] Jan J. M. M. Rutten. "Algebraic Specification and Coalgebraic Synthesis of Mealy Automata". In: *Proceedings of the International Workshop on Formal Aspects of Component Software, FACS 2005, Macao, October 24-25, 2005*. Ed. by Zhiming Liu and Luís Soares Barbosa. Vol. 160. Electronic Notes in Theoretical Computer Science. Elsevier, 2005, pp. 305–319. DOI: 10.1016/j.entcs.2006.05.030. URL: https://doi.org/10.1016/j.entcs.2006.05.030.

[37] Jan J. M. M. Rutten. "Universal coalgebra: a theory of systems". In: *Theor. Comput. Sci.* 249.1 (2000), pp. 3–80. DOI: 10.1016/S0304-3975(00)00056-6. URL: https://doi.org/10.1016/S0304-3975(00)00056-6.

[38] Davide Sangiorgi. *Introduction to Bisimulation and Coinduction*. Cambridge University Press, 2011.

[39] André Schiper, Jorge Eggli, and Alain Sandoz. "A New Algorithm to Implement Causal Ordering". In: *Proceedings of the 3rd International Workshop on Distributed Algorithms*. Berlin, Heidelberg: Springer-Verlag, 1989, pp. 219–232. ISBN: 3540516875.

[40] Frank B Schmuck. "The use of efficient broadcast protocols in asynchronous distributed systems". PhD thesis. Cornell University, 1988.

[41] Fred B. Schneider. "Implementing Fault-Tolerant Services Using the State Machine Approach: A Tutorial". In: *ACM Comput. Surv.* 22.4 (1990), pp. 299–319. ISSN: 0360-0300. DOI: 10.1145/98163.98167. URL: https://doi.org/10.1145/98163.98167.

[42] Reinhard Schwarz and Friedemann Mattern. "Detecting causal relationships in distributed computations: In search of the holy grail". In: *Distributed computing* 7.3 (1994), pp. 149–174.

[43] Marc Shapiro et al. "Conflict-Free Replicated Data Types". In: *Stabilization, Safety, and Security of Distributed Systems*. Ed. by Xavier Défago, Franck Petit, and Vincent Villain. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 386–400. ISBN: 978-3-642-24550-3.

[44] Amin Timany et al. "Trillium: Higher-Order Concurrent and Distributed Separation Logic for Intensional Refinement". In: *Proc. ACM Program.*

Lang. 8.POPL (2024). DOI: 10.1145/3632851. URL: https://doi.org/10.1145/3632851.

[45] Peter Zeller, Annette Bieniusa, and Arnd Poetzsch-Heffter. "Formal Specification and Verification of CRDTs". In: *Formal Techniques for Distributed Objects, Components, and Systems*. Ed. by Erika Ábrahám and Catuscia Palamidessi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 33–48. ISBN: 978-3-662-43613-4.