CRDTs, Coalgebraically

Nathan Liittschwager
University of California, Santa Cruz, CA, USA

Stelios Tsampas
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Jonathan Castello
University of California, Santa Cruz, CA, USA

Lindsey Kuper
University of California, Santa Cruz, CA, USA

Abstract

We describe ongoing work that models conflict-free replicated data types (CRDTs) from a coalgebraic point of view. CRDTs are data structures designed for replication across multiple physical locations in a distributed system. We show how to model a CRDT at the local replica level using a novel coalgebraic semantics for CRDTs. We believe this is the first step towards presenting a unified theory for specifying and verifying CRDTs and replicated state machines. As a case study, we consider *emulation* of CRDTs in terms of coalgebra.

2012 ACM Subject Classification Theory of computation; Theory of computation → Semantics and reasoning; Theory of computation → Distributed computing models; Theory of computation → Concurrency; Theory of computation → Formal languages and automata theory

Keywords and phrases Coalgebra, Distributed Systems, Concurrency, Bisimulation

Digital Object Identifier 10.4230/LIPIcs.CALCO.2023.22

Category Early Ideas

Funding This material is based upon work supported by the National Science Foundation under Grant No. CCF-2145367. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

1 Introduction

In distributed systems, data replication guards against machine failures and keeps data physically close to clients who require low-latency access, but it introduces the problem of keeping replicas consistent with one another in the face of network partitions and unpredictable message latency. *Conflict-free replicated data types* (CRDTs) [14, 13] are data structures whose operations must satisfy certain mathematical properties that can be leveraged to ensure *strong convergence*, meaning that replicas are guaranteed to have equivalent state given that they have received and applied the same (unordered) set of update operations.

A typical use case for CRDTs is that of a text document being made available to n concurrent users via an online collaborative editor. We can model such a collaborative editing application as a coalgebra

$$\langle \mathsf{update}, \mathsf{query} \rangle : X \to X^A \times S,$$

where X is some black-box state (representing the editor internals), A a finite set of arguments, e.g, add or delete a character at a particular location via $\mathsf{update} : X \to X^A$, and S being the *observable* state: the text string itself accessible through $\mathsf{query} : X \to S$. As a shorthand, we use u and q for update and query, respectively.

Viewing each client as acting on the text document via $u : X \to X^A$, we can define $u^* : X \to X^{A^*}$ by recursion as an iterated update that consumes a *list* of commands $\sigma \in A^*$ and gives the current state: $u^*(x)(\langle\rangle) = x$ and $u^*(x)(a \cdot \sigma) = u^*(u(x)(a))(\sigma)$. Hence we may understand that all client interactions are represented as words in $\sigma \in A^*$, and the state of the object $x \in X$ is generated by $u^*(x)(\sigma)$. Observe that the coalgebra $\langle u, q\rangle : X \to X^A \times S$ is a $\big((-)^A \times S\big)$-coalgebra, and hence by standard results [4]:

▶ **Proposition 1.** *The final $\big((-)^A \times S\big)$-coalgebra is given by*

$$S^{A^*} \xrightarrow{\langle \zeta_1, \zeta_2\rangle} \left(S^{A^*}\right)^A \times S,$$

where $\zeta_1(\varphi)(a) = \lambda(\sigma \in A^*).\ \varphi(a{\cdot}\sigma)$ and $\zeta_2(\varphi) = \varphi(\langle\rangle)$. That is, for any $\big((-)^A \times S\big)$-coalgebra $\langle u, q\rangle : X \to X^A \times S$, there is a unique coalgebra homomorphism $beh : X \to S^{A^*}$ that satisfies $beh(x) = \lambda(\sigma \in A^*).\ q(u^*(x)(\sigma))$.

The final coalgebra $\langle \zeta_1, \zeta_2\rangle : S^{A^*} \to \left(S^{A^*}\right)^A \times S$ thus defines all possible infinite behaviors of the collaborative text editor, under the assumption that it is implemented by some centralized server that totally orders and executes all client requests. However, such a centralized approach is often infeasible or simply undesirable. Instead, each client may be working on a local copy – a *replica* – of the object and propagating changes between replicas.

To keep replicas consistent, specialized algorithms and communication middleware can be used. For example, the *state machine replication* [11] approach guarantees *strong consistency* (informally, where clients cannot tell that the data is replicated) by ensuring that each replica executes the same sequence of commands. However, this approach requires expensive coordination between replicas. CRDTs take a different approach: they avoid the need for coordination by carefully constraining the state space X and the implementation of the update and query methods, and sacrifice strong consistency in favor of *strong convergence*. Under strong convergence, replicas that have received the same set of updates (in any order) agree in state, but clients may observe differing intermediate states.

In this short paper, we propose studying CRDTs under a coalgebraic lens. We find that CRDTs lend themselves to a coalgebraic interpretation: they are implemented as replicated objects at multiple locations, where each replica has an opaque internal state, but publicly available methods centered around calls to *update* or *query*; strong convergence and emulation of CRDTs are primarily about *observable behavior* from perspective of a client. By taking the coalgebraic approach, we can use the well-developed theory of universal coalgebra to reason about strong convergence of CRDTs. Moreover, the coalgebraic approach lets us make precise a notion of *emulation* of CRDTs that has until now been known only informally.

2 Coalgebraic Semantics of CRDTs

CRDTs may be specified in an *operation-based* (or *op-based*) style or in a *state-based* style. Op-based CRDTs require that replicas transmit messages containing the *effects* of a local update *downstream* to other replicas. Updates are *applied* to replicas in a way that respects causality [5]: when an update is applied to replica i, any causally preceding updates must have already been applied to i, but *concurrently* applied updates do not need to be restricted to any particular order; an op-based CRDT converges regardless. State-based CRDTs, on the other hand, converge by restricting their (observable) state to be a join-semilattice $\langle S, \sqcup\rangle$, and updates are propagated between replicas by simply passing copies of state $s \in S$

between replicas, and merging states via a least-upper-bound operator $\sqcup$. The two styles are equivalent in the sense that given a state-based CRDT, one can construct a corresponding op-based CRDT that *emulates* it, and vice versa [14].

We show that the semantics of a CRDT can be described as a coalgebra $c : X \to F(X)$, where $F : \mathbf{Sets} \to \mathbf{Sets}$ is a Kripke-polynomial functor [4], beginning with the semantics for state-based CRDTs. Similarly to the text-editing example, the CRDT consists of a state space X, a set S of observables, an update map $u : X \to X^A$ and a query map $q : X \to S$. In addition, we assume an abstract set of *events* E and equip the CRDT with a "history" morphism $h : X \to \mathcal{P}(E)$, interpreted as a kind of log for events $e \in E$ that have happened at the replica. Intuitively, events are given by a map that "wraps" interactions (e.g., inputs $a \in A$) with the environment and tag it with meta-data, such as a sequence number.

To model the least-upper-bound operator $\sqcup$, we require a map $\mathsf{merge} : X \to X^S$ that allows an object with state x to receive an input state $s = q(x') \in S$ from some other replica x', along with rules requiring that merge is inflationary wrt to queries. *Strong convergence* is defined as the property $\forall x, x' \in X.(h(x) = h(x') \implies q(x) = q(x'))$, which CRDT coalgebras must satisfy: when two replica states have observed the same set of events, then their query state is the same.

▶ **Definition 2.** *A state-based CRDT consists of a state space X, inputs A, events E, a payload S where $S = \langle S, \sqcup\rangle$ is a join-semilattice, and maps*

$$\langle u, q, h, \xi, \mathsf{merge}\rangle : X \to X^A \times S \times \mathcal{P}(E) \times \mathcal{P}(E)^{A+S} \times X^S,$$

s.t. the following hold for all $x \in X$, $s \in S$, $a \in A$,

(i) $q(\mathsf{merge}(x)(s)) = q(x) \sqcup s$*;*
(ii) $q(x) \sqcup q(u(x)(a)) = q(u(x)(a))$*;*
(iii) $h(u(x)(a)) = h(x) \cup \xi(x)(a)$*;*
(iv) $h(\mathsf{merge}(x)(s)) = h(x) \cup \xi(x)(s)$*;*
(v) $\forall x' \in X.\ (h(x) = h(x') \implies q(x) = q(x'))$*.*

Op-based CRDTs are similar, except they define local updates $u : X \to X^A$ in terms of two other methods: a side-effect free *prepare* method $\mathsf{prep} : X \to M^A$ and an effectful *apply* method $\mathsf{app} \to X^M$, where M is a set of messages. We assume M is equipped with a partial order $\prec_{hb}$, the so-called *happens-before* relation [5, 12]. This implies that M is equipped with metadata sufficient for $\prec_{hb}$ to make sense. Say elements m, m' are *concurrent* and write $m \parallel m'$ if $\neg((m \prec_{hb} m') \vee (m' \prec_{hb} m))$. Upon a client invoking a local update of type $a \in A$, op-based CRDTs first generate a message $m \in M$ with prep, and then send m downstream to neighboring replicas using some communication middleware that ensures that messages are delivered in an order consistent with causality [12]. The middleware delays delivery of m to a replica with state x until a decider (assumed sufficient to ensure causality) $\mathsf{dlvr?} : X \to \mathbf{2}^M$ returns "yes", after which the message is applied with app. The decider $\mathsf{dlvr?}$ can be implemented independently of the CRDT application. A common approach is the vector clock protocol [8, 2].

▶ **Definition 3.** *An operation-based CRDT consists of a state space X, inputs A, events E, payload S, and maps*

$$\langle u, q, h, \xi, \mathsf{dlvr?}, \mathsf{prep}, \mathsf{app}\rangle : X \to X^A \times S \times \mathcal{P}(E)^M \times \mathcal{P}(E)^E \times \mathbf{2}^E \times E^A \times X^E$$

s.t. the following hold for all $x \in X$, $a \in A$, $m, m' \in M$

(i) $u(x) = \lambda(a \in A).\mathsf{app}(x)(\mathsf{prep}(x)(a))$*;*
(ii) $h(u(x)(a)) = h(x) \cup \xi(x)(\mathsf{prep}(x)(a))$*;*

(iii) $h(\mathsf{app}(x)(m)) = h(x) \cup \xi(x)(m)$
(iv) *if* $\mathsf{dlvr?}(x)(m) = \mathsf{dlvr?}(x)(m') = \top$, *then updates* m, m commute. *I.e.,* $q(x') = q(x'')$ *where* $x' = \mathsf{app}(\mathsf{app}(x)(m))(m')$ *and* $x'' = \mathsf{app}(\mathsf{app}(x)(m'))(m)$.
(v) $\forall x' \in X.\ (h(x) = h(x') \implies q(x) = q(x'))$

Critically, the coalgebra above models a *single* replica, of which there are n many, initialized from some starting state $x_0 \in X$. Communication, from this point of view, is abstracted to the communication middleware.

3 Emulation of CRDTs

Much existing work on CRDT semantics (e.g., [1, 3, 9, 7, 6, 10]) has treated op-based and state-based CRDTs as distinct classes of objects, often only considering one class or the other. The justification for this approach is that a state-based CRDT can *emulate* a corresponding op-based CRDT, and vice versa [14]. Despite this commonly cited fact, a notion of emulation is never made precise. Here we aim to fill this gap by showing that emulation of CRDTs may be thought of in terms of *bisimulation* of transition systems.

▶ **Definition 4.** *A* transition system *on a state space* X *with observations* S *is a coalgebra* $\langle \mathsf{next}, \mathsf{obs}\rangle : X \to \mathcal{P}(X) \times S$ *s.t.* $x \longrightarrow x' \iff x' \in \mathsf{next}(x)$, *and* $x \downarrow s \iff s = \mathsf{obs}(x)$. *Given two transition systems* $\langle \mathsf{next}_1, \mathsf{obs}_1\rangle : X \to \mathcal{P}(X) \times S$ *and* $\langle \mathsf{next}_2, \mathsf{obs}_2\rangle : Y \to \mathcal{P}(Y) \times S$, *a relation* $R \subseteq X \times Y$ *is a* bisimulation *iff* $\forall (x, y) \in X \times Y$, *if* $R(x, y)$, *then*

- $x \downarrow s \implies y \downarrow s$;
- $x \longrightarrow x' \implies \exists y'.\ y \longrightarrow y'$;
- $y \longrightarrow y' \implies \exists x'.\ x \longrightarrow x'$.

It can be shown [4] that coalgebras $c : X \to F(X)$ may be mapped to transition systems $d : X \to \mathcal{P}(X)$.For the coalgebra of definition 2, define $x \longrightarrow x' \iff (\exists a \in A.\ u(x)(a) = x') \vee (\exists s \in S.\ \mathsf{merge}(x)(s) = x')$, and $x \downarrow s \iff q(x) = s$. The construction for definition 3 is similar, with the restriction that $x \longrightarrow x' \iff (\exists a \in A.\ u(x)(a) = x') \vee (\exists m \in M.\ \mathsf{app}(x)(m) = x' \wedge \mathsf{dlvr?}(x)(m))$.

This translation to transition systems exposes the similarity of state-based and operation-based CRDTs by revealing there are really only two "kinds" of transition steps: local steps via $u : X \to X^A$ and synchronization steps via $\mathsf{merge} : X \to X^S$ for state-based CRDTs, and $\mathsf{app} : X \to X^M$ for operation-based CRDTs. For both kinds of CRDTs, the observable payload is given by $x \downarrow s$.

▶ **Proposition 5** (Emulation of state-based CRDTS by op-based CRDTs)**.** *Let* $F, G : \mathbf{Sets} \to \mathbf{Sets}$ *be appropriate functors s.t. given coalgebra* $X \xrightarrow{\langle u,q,h,\xi,\mathsf{merge}\rangle} F(X)$ *satisfying definition 2, with transition system semantics* $X \xrightarrow{\langle \mathsf{next}_1,\mathsf{obs}_1\rangle} \mathcal{P}(X) \times S$. *Then there is a coalgebra* $Y \xrightarrow{\langle u',q',h',\xi',\mathsf{dlvr?},\mathsf{prep},\mathsf{app}\rangle} G(Y)$ *satisfying definition 3 with transition semantics* $Y \xrightarrow{\langle \mathsf{next}_2,\mathsf{obs}_2\rangle} \mathcal{P}(Y) \times S$ *s.t. there is a bisimulation relation* $R : X \times Y$ *between* $X \xrightarrow{\langle \mathsf{next}_1,\mathsf{obs}_1\rangle} \mathcal{P}(X) \times S$ *to* $Y \xrightarrow{\langle \mathsf{next}_2,\mathsf{obs}_2\rangle} \mathcal{P}(Y) \times S$.

4 Future Work

There are two main directions for future work.

First, the semantics given here can be lifted to consider the semantics of CRDTs (and possible state machine replication in general) from a more *global* point of view, i.e., as interacting asynchronous processes.

Second, the above proposition only gives one direction of the emulation result. The other direction is left to future work. More generally, both operation-based and state-based CRDTs need exhibit strong convergence, which can be thought of as a form of observational equivalence, similar to how emulation is approached here. However, a more interesting approach might be to frame both operation-based and state-based CRDTs as satisfying strong convergence as a universal property, showing that the difference between CRDTs amounts to nothing more than choice of construction.

References

1. Sebastian Burckhardt, Alexey Gotsman, Hongseok Yang, and Marek Zawirski. Replicated data types: Specification, verification, optimality. In *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '14, pages 271–284, New York, NY, USA, 2014. Association for Computing Machinery. `doi:10.1145/2535838.2535848`.
2. C. J. Fidge. Timestamps in message-passing systems that preserve the partial ordering. *Proceedings of the 11th Australian Computer Science Conference*, 10(1):56–66, 1988.
3. Victor B. F. Gomes, Martin Kleppmann, Dominic P. Mulligan, and Alastair R. Beresford. Verifying strong eventual consistency in distributed systems. *Proc. ACM Program. Lang.*, 1(OOPSLA), October 2017. `doi:10.1145/3133933`.
4. Bart Jacobs. *Introduction to Coalgebra: Towards Mathematics of States and Observation*, volume 59 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2016. `doi:10.1017/CBO9781316823187`.
5. Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, July 1978. `doi:10.1145/359545.359563`.
6. 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, pages 636–650, New York, NY, USA, 2021. Association for Computing Machinery. `doi:10.1145/3453483.3454067`.
7. Yiyun Liu, James Parker, Patrick Redmond, Lindsey Kuper, Michael Hicks, and Niki Vazou. Verifying replicated data types with typeclass refinements in Liquid Haskell. *Proc. ACM Program. Lang.*, 4(OOPSLA), November 2020. `doi:10.1145/3428284`.
8. Friedemann Mattern. Virtual time and global states of distributed systems. In *Parallel and Distributed Algorithms*, pages 215–226. North-Holland, 1989.
9. Kartik Nagar and Suresh Jagannathan. Automated parameterized verification of CRDTs. In Isil Dillig and Serdar Tasiran, editors, *Computer Aided Verification*, pages 459–477, Cham, 2019. Springer International Publishing.
10. Abel Nieto, Léon Gondelman, Alban Reynaud, Amin Timany, and Lars Birkedal. Modular verification of op-based CRDTs in separation logic. *Proc. ACM Program. Lang.*, 6(OOPSLA2), October 2022. `doi:10.1145/3563351`.
11. Fred B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Comput. Surv.*, 22(4):299–319, December 1990. `doi:10.1145/98163.98167`.
12. Reinhard Schwarz and Friedemann Mattern. Detecting causal relationships in distributed computations: In search of the holy grail. *Distributed Computing*, 7(3):149–174, March 1994. `doi:10.1007/BF02277859`.
13. Marc Shapiro, Nuno Preguiça, Carlos Baquero, and Marek Zawirski. A comprehensive study of Convergent and Commutative Replicated Data Types. Research Report RR-7506, Inria – Centre Paris-Rocquencourt; INRIA, January 2011. URL: `https://hal.inria.fr/inria-00555588`.
14. Marc Shapiro, Nuno Preguiça, Carlos Baquero, and Marek Zawirski. Conflict-free replicated data types. In Xavier Défago, Franck Petit, and Vincent Villain, editors, *Stabilization, Safety, and Security of Distributed Systems*, pages 386–400, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.