CMCS 2024

Coinductive Reasoning about CRDT Emulation

Nathan E. Liittschwager,
Stelios Tsampas,
Jonathan Castello,
Lindsey Kuper

Background: State Machine Replication

Imagine a *service* offered to a *client* in the form of a black-box with I/O behavior and internal state

The client inputs are *requests* in the form of *commands* $a \in A$ which update the internal state.

Client observations $b \in B$ are accessible by a *query* method call

Implement the service as a *state machine* (coalgebra):

$$(u, q) : X \to X^A \times B$$

Background: State Machine Replication

To achieve *fault tolerance* the state machine $(u, q) : X \to X^A \times B$ is *replicated* on n servers or nodes: $(u_1, q_1), \ldots, (u_n, q_n) : X \to X^A \times B$ with same initial state $x \in X$

Client input seen as a totally ordered sequence $(a_1, \ldots, a_k) \in A*$

Each replica $(x \in X, (u_i, q_i))$ must compute the same sequence $(a_1, \ldots, a_k) \in A*$, thus obtain the same state *(linearizability)*

Fault tolerance achieved: if one replica goes down, another replica can take its role

Background: State Machine Replication

To achieve *fault tolerance* the state machine $(u, q) : X \to X^A \times B$ is *replicated* on n servers or nodes: $(u_1, q_1), \ldots, (u_n, q_n) : X \to X^A \times B$ with same initial state $x \in X$

Client input seen as a totally ordered sequence $(a_1, \ldots, a_k) \in A*$

Each replica $(x \in X, (u_i, q_i))$ must compute the same sequence $(a_1, \ldots, a_k) \in A*$, thus obtain the same state *(linearizability)*

Fault tolerance achieved: if one replica goes down, another replica can take its role

Background: State Machine Replication

To achieve *fault tolerance* the state machine $(u, q) : X \to X^A \times B$ is *replicated* on n servers or nodes: $(u_1, q_1), \ldots, (u_n, q_n) : X \to X^A \times B$ with same initial state $x \in X$

Client input seen as a totally ordered sequence $(a_1, \ldots, a_k) \in A*$

Non-trivial!

Each replica $(x \in X, (u_i, q_i))$ must compute the same sequence $(a_1, \ldots, a_k) \in A*$, thus obtain the same state *(linearizability)*

Fault tolerance achieved: if one replica goes down, another replica can take its role

Background: CRDTs

Totally ordering commands $(a_1, \ldots, a_k) \in A^*$ non-trivial in asynchronous distributed systems

Conflict-free Replicated Data Types (CRDTs) solve this problem by by using data structures which don't require total order

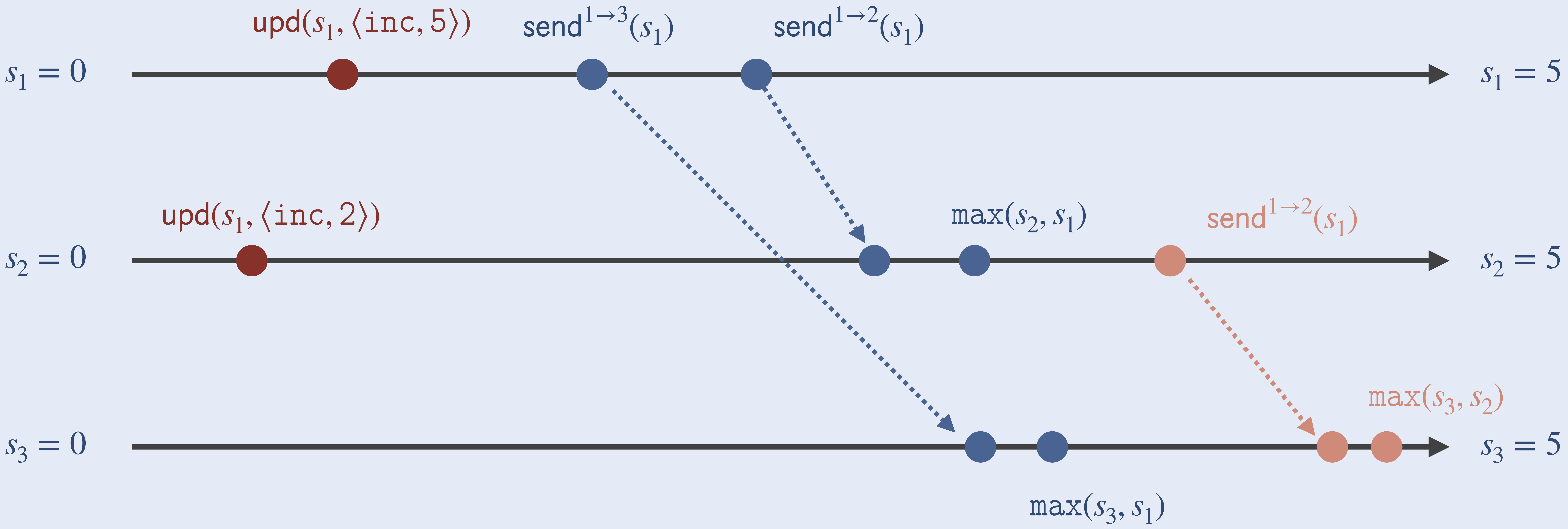
Two major flavors of CRDT: "State-based" and "Operation-based"

Both achieve *strong convergence* - if they know about the same set of messages, they have the same state.

Background: State-Based CRDTs

State-based CRDTs use a join-semilattice $(S, \sqcup)$ as the state-space

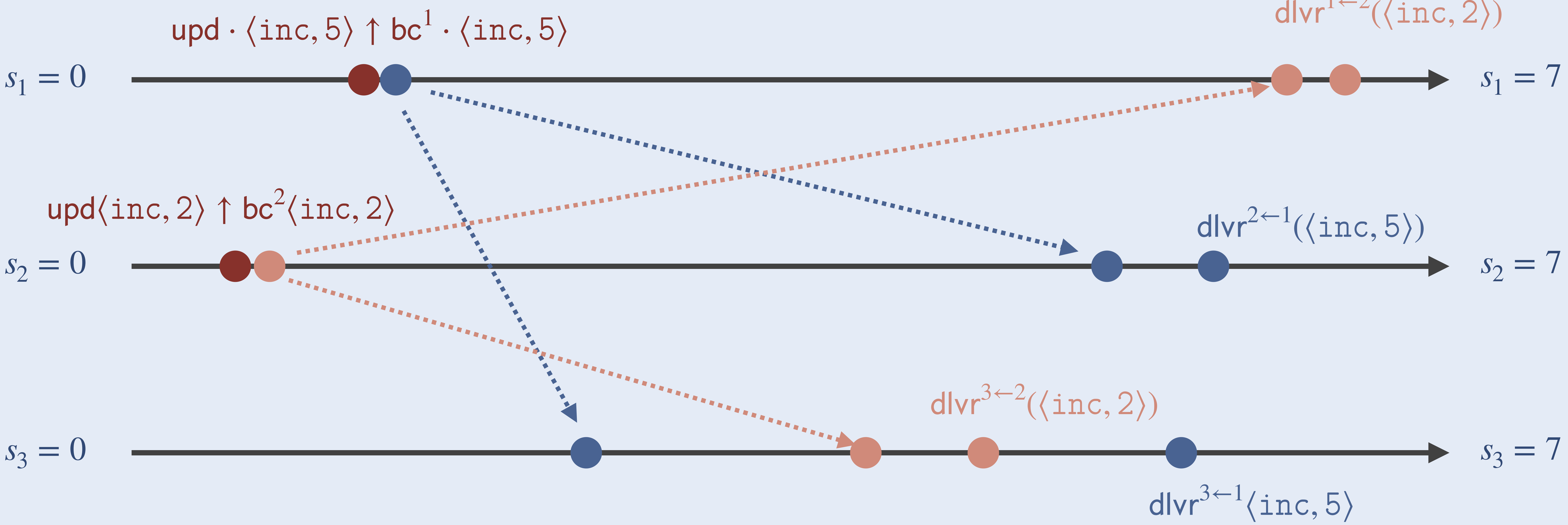
Updates must be *inflationary*: $s \sqcup \mathsf{upd}(s, a) = \mathsf{upd}(s, a)$ for all $s \in S, a \in A$



Background: Operation-based CRDTs

Operation-based CRDTs require messages (operations) be a *partial order* $(M, \prec)$

Each replica executes the same set of messages in a way consistent with $\prec$, communication is by *broadcast*



Background: CRDT Emulation and Equivalence

State-based and op-based CRDTs are often considered to be *equivalent*.

The reasoning is that they "*emulate*" each other: there are a pair of maps $\mathscr{F}, \mathscr{G}$ to translate between the two types [Shapiro et al. 2011]

Definition (Emulation Maps)

State-based CRDT: $((S, \sqcup), s_0, \mathtt{u}, \mathtt{q})$ Op-based CRDT: $(S, s_0, M, \mathtt{u}, \mathtt{t}, \mathtt{e}, \mathtt{q})$

effect-msg

prepare-msg

$$((S, \sqcup), s_0, \mathtt{u}, \mathtt{q}) \overset{\mathscr{F}}{\mapsto} (S, s_0, S, \mathtt{u}, \mathtt{u}, \sqcup, \mathtt{q})$$

$$(S, s_0, M, \mathtt{u}, \mathtt{t}, \mathtt{e}, \mathtt{q}) \overset{\mathscr{G}}{\mapsto} ((\mathscr{P}_{fin}(M), \cup), \emptyset, \mathtt{u}', \mathtt{q}')$$

where $\mathtt{u}'(H, a) = H \cup \{\mathtt{t}([\![H]\!], a)\}$, and $\mathtt{q}'(H) = \mathtt{q}([\![H]\!])$ and $[\![\cdot]\!] : \mathscr{P}_{fin}(M) \to S$ is appropriate map to translate sets of messages into state

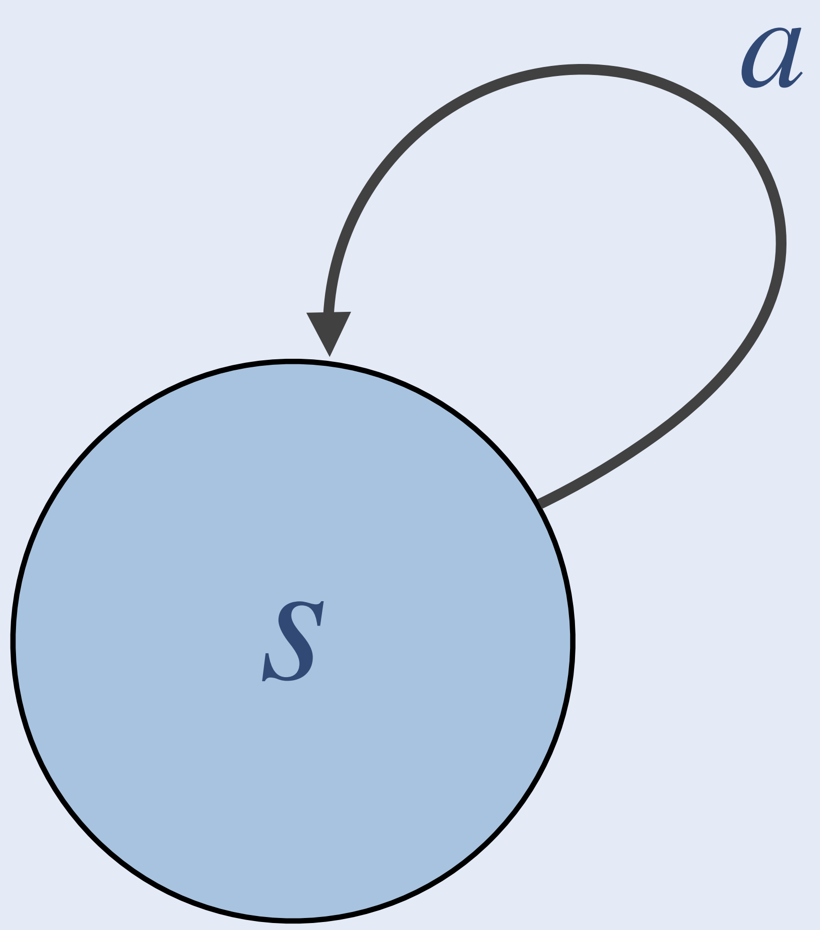
Background: CRDT Emulation and Equivalence

Maps $\mathscr{F}$ and $\mathscr{G}$ are intuitively, potentially correct

But, emulation is not rigorously defined: no formal requirements on behavior, only informal arguments about strong eventual convergence

What if we defined $\mathscr{F}$ to map each replica to a trivial state machine?

$$S = \{s\}$$
$$\forall a \in A \,.\, u(s)(a) = s$$
$$q(s) = \top$$



Convergence!

Strong Bisimulation?

What if we required the original CRDT and the $\mathscr{F}$-emulator (or $\mathscr{G}$-emulator) to exhibit a bisimulation?

For example, if $\mathscr{G}$ were a coalgebra homomorphism - Very strong notion of “equivalence”!

But is there even such a bisimulation?

Strong Bisimulation?

What if we required the original CRDT and the $\mathscr{F}$-emulator (or $\mathscr{G}$-emulator) to exhibit a bisimulation?

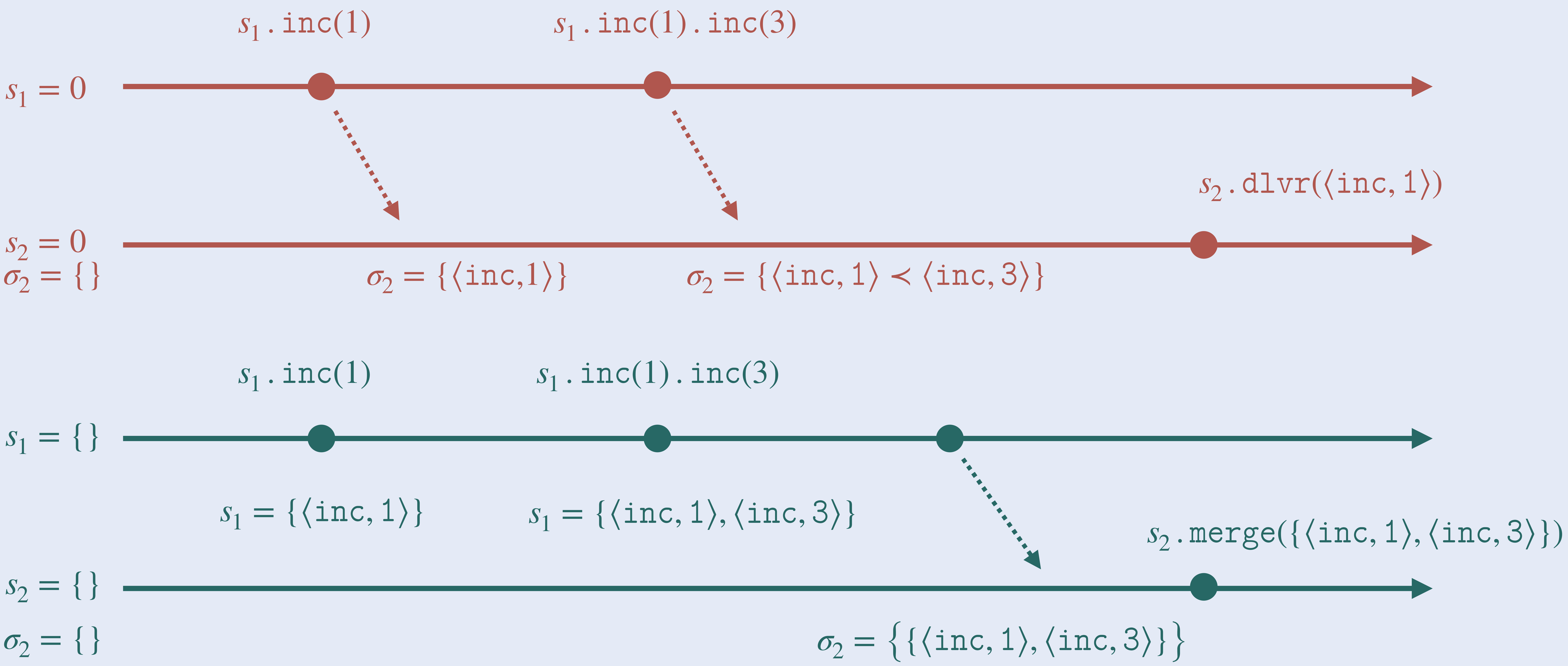
For example, if $\mathscr{G}$ were a coalgebra homomorphism - Very strong notion of "equivalence"!

But is there even such a bisimulation? NO.

The semantics of Op-based CRDTs treat $\mathsf{upd}(a) \uparrow \mathsf{bc}^i(m)$ events as *atomic* (uninterruptible)

But there is **no** such requirement for $\mathsf{upd}(a)$ and $\mathsf{send}^{i \to j}(s)$ events on state-based CRDTs

Bisimulation Game



Bisimulation Game

Let $[\![\cdot]\!] : \mathscr{P}_{fin}(M) \to S$ be a map which interprets sets of messages to states

Define the query map $q_{\mathsf{op}} = id_S$

Define the query map $q_{\mathsf{st}} = id_S \circ [\![\cdot]\!]$

$$s_1 \,.\, \mathtt{inc}(1) \,.\, \mathtt{inc}(3) \implies q(s_1) = 4$$

$$s_2 \,.\, \mathtt{dlvr}(\langle \mathtt{inc}, 1 \rangle) \implies q(s_2) = 1$$

$$s_1 \,.\, \mathtt{inc}(1) \,.\, \mathtt{inc}(3) \implies q'(s_1) = [\![\{ \langle \mathtt{inc}, 1 \rangle, \langle \mathtt{inc}, 3 \rangle \}]\!] = 4$$

$$s_2 \,.\, \mathtt{merge}(\{ \langle \mathtt{inc}, 1 \rangle, \langle \mathtt{inc}, 3 \rangle \}) \implies q'(s_2) = [\![\{ \langle \mathtt{inc}, 1 \rangle, \langle \mathtt{inc}, 3 \rangle \}]\!] = 4$$

Background: CRDT Emulation and Equivalence

Despite this, the notion of emulation is "load bearing" in the CRDT literature

> *"...our techniques [on op-based CRDTs] naturally extends to state-based CRDTs since they can be emulated by an op-based model..."* - [Nagar et al., 2019]

> *"... [our work on synthesis of state-based CRDTs]... can always be translated to op-based CRDTs if necessary..."* - [Laddad et al., 2022]

Our contributions: We close this gap by showing that $\mathscr{F}$ and $\mathscr{G}$ induce a pair of *weak simulations* between the original CRDT and its "emulator"

Definition (Weak Simulation)

Let $(X, (h, \mathsf{obs}_1))$ and $(Y, (g, \mathsf{obs}_2))$ be coalgebras of endofunctor $\mathscr{P}(\,-\,) \times B$ and let $g* : Y \to \mathscr{P}(Y)$ be the *reflexive, transitive closure* of g.

A *weak simulation* of (X, h) and (Y, g) is a relation $R \subseteq X \times Y$ s.t.
$\forall (x, y) \in X \times Y.$ if $(x, y) \in R$, then

1. $\mathsf{obs}_1(x) = b \implies \mathsf{obs}_2(y) = b$
2. $x' \in h(x) \implies \exists y' \in g*(y) \wedge (x', y') \in R$

Coinductive Reasoning about CRDT Emulation

Definition (Op-based CRDT Systems)

$$\frac{\alpha \notin \texttt{update} \quad x_j \longrightarrow_{\mathsf{op}} x_j' \uparrow (a, m)}{\langle \alpha, (x_i)_{i \in n} \rangle \rightsquigarrow_{\mathsf{op}} \langle \texttt{upd}^j(a, m), (x_i)_{i \in n}[x_j \leftarrow x_j'] \rangle} \text{[OpUpdate]}$$

$$\frac{(x_i')_{i \in n} = \texttt{bcast}^j_m (x_i)_{i \in n}}{\langle \mathsf{upd}^j(a, m), (x_i)_{i \in n} \rangle \rightsquigarrow_{\mathsf{op}} \langle \mathsf{bc}^j(m), (x_i')_{i \in n} \rangle} \text{[OpBroadcast]}$$

$$\frac{\alpha \notin \texttt{update} \quad x_j \longrightarrow_{\mathsf{op}} x_j' \text{ via deliver } m}{\langle \alpha, (x_i)_{i \in n} \rangle \rightsquigarrow_{\mathsf{op}} \langle \mathsf{dlvr}^j(m), (x_i)_{i \in n}[x_j \leftarrow x_j'] \rangle} \text{[OpDeliver]}$$

(Events: $\alpha := \top \mid \mathsf{upd}^i(a, m) \mid \mathsf{bc}^i(m) \mid \mathsf{dlvr}^i(m)$)

Coinductive Reasoning about CRDT Emulation

Definition (State-based CRDT Systems)

$$\frac{x_j = (s_j, \sigma_j) \quad x_j \longrightarrow_{\mathsf{st}} x'_j \quad (\mathrm{u}(s_j, a), \sigma_j) = x'_j}{\langle \alpha, (x_i)_{i \in n} \rangle \rightsquigarrow_{\mathsf{st}} \langle \mathsf{upd}^j(a), (x_i)_{i \in n}[x_j \leftarrow x'_j] \rangle} \quad \text{[StUpdate]}$$

$$\frac{x_i = (s_i, \sigma_i) \quad x_j = (s_j, \sigma_j) \quad x'_j = (s_j, \sigma_j \cup \{s_i\})}{\langle \alpha, (x_i)_{i \in n} \rangle \rightsquigarrow_{\mathsf{st}} \langle \mathsf{send}^{i \to j}(s_i), (x_i)_{i \in n}[x_j \leftarrow x'_j] \rangle} \quad \text{[StSend]}$$

$$\frac{x_j = (s_j, \sigma_j) \quad x_j \longrightarrow_{\mathsf{st}} x'_j \quad (s_j \sqcup s, \sigma_j \backslash \{s\}) = x'_j}{\langle \alpha, (x_i)_{i \in n} \rangle \rightsquigarrow_{\mathsf{st}} \langle \mathsf{dlvr}^j(s), (x_i)_{i \in n}[x_j \leftarrow x'_j] \rangle} \quad \text{[StDeliver]}$$

(Events: $\alpha := \top \mid \mathsf{upd}^i(a) \mid \mathsf{send}^{i \to j}(s) \mid \mathsf{dlvr}^i(s)$)

Weak Simulation (State-based $\rightarrow$ Op-based)

Theorem (Weak Simulation)

Let $(\rightsquigarrow_{\mathsf{st}}, q_{\mathsf{st}})$ be the state-based CRDT system for $c = ((S, \sqcup), s_0, u, q)$ and $(\rightsquigarrow_{\mathsf{op}}, q_{\mathsf{op}})$ the op-based *emulator* CRDT system for $\mathscr{F}(c)$. There are a pair of weak simulations Q_1 and Q_2 such that,

1. Q_1 is a weak simulation for $(\rightsquigarrow_{\mathsf{op}}, q_{\mathsf{op}})$ and $(\rightsquigarrow_{\mathsf{st}}, q_{\mathsf{st}})$
2. Q_2 is a weak simulation for $(\rightsquigarrow_{\mathsf{st}}, q_{\mathsf{st}})$ and $(\rightsquigarrow_{\mathsf{op}}, q_{\mathsf{op}})$

Weak Simulation (Op-based $\rightarrow$ State-based)

Theorem (Weak Simulation)

Let $(\rightsquigarrow_{\text{op}}, q_{\text{op}})$ be the op-based CRDT system for $o = (S, s_0, M, u, t, e, q)$ and $(\rightsquigarrow_{\text{st}}, q_{\text{st}})$ the state-based *emulator* CRDT system for $\mathcal{G}(o)$. There are a pair of weak simulations R_1 and R_2 such that,

1. R_1 is a weak simulation for $(\rightsquigarrow_{\text{op}}, q_{\text{op}})$ and $(\rightsquigarrow_{\text{st}}, q_{\text{st}})$
2. R_2 is a weak simulation for $(\rightsquigarrow_{\text{st}}, q_{\text{st}})$ and $(\rightsquigarrow_{\text{op}}, q_{\text{op}})$

Future Work

- It would be interesting to try to capture this notion of *emulation* (translation + simulation) in higher generality. Perhaps in different categories (e.g., Kleisli), perhaps with different systems.

- Our model is based on more classical distributed systems theory ("vectors" of transition systems) but is nonetheless coalgebraic. A general coalgebraic treatment of distributed systems theory would be interesting - the challenge is reconciling the different notions of "simulation" under coalgebraic lens

- What are the interesting properties preserved by a translation + simulation?

References

1. 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.
2. 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.
3. 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.

Questions?