More Conflict-Free Replicated Data Types

Lattice Theory for Parallel Programming

Pierre Talbot

pierre.talbot@uni.lu

17th October 2025

University of Luxembourg

Definition of State-based CRDT

Definition

A state-based conflict-free replicated data type is a tuple $\langle L, \leq, S, \mathit{value}, f_1, \ldots, f_n \rangle$ where:

- $\langle L, \leq, \sqcup \rangle$ is a lattice[1].
- S is a set of values.
- $f_1, \ldots, f_n$ are monotone and extensive functions over L (extensive: $\forall x, x \leq f(x)$).
- $\mathit{value} : L \rightarrow S$ returns the value modeled by the CRDT.

[1] A join semi-lattice to be precise, because we don't need the meet operation $\sqcap$.

Grow-only Counter

G-Counter

Let n be the number of replicas (nodes of the distributed system).
G-Counter is a CRDT $GC \triangleq \langle \mathbb{Z}^n, \dot{\leq}, \dot{\sqcup}, \bot, \mathbb{Z}, value, increment_1, \ldots, increment_n \rangle$, where:

- $(x_1, \ldots, x_n) \mathrel{\dot{\leq}} (y_1, \ldots, y_n) \Leftrightarrow \forall 1 \leq i \leq n, x_i \leq y_i$.
- $(x_1, \ldots, x_n) \mathbin{\dot{\sqcup}} (y_1, \ldots, y_n) \triangleq (max(x_1, y_1), \ldots, max(x_n, y_n))$.
- $\bot = (0, \ldots, 0)$.
- $increment_k(\langle x_1, \ldots, x_n \rangle) \triangleq \langle x_1, \ldots, x_k + 1, \ldots, x_n \rangle$.
- $value(\langle x_1, \ldots, x_n \rangle) \triangleq \sum_{1 \leq i \leq n} x_i$.

where $increment_k$ is executed on the k^{th} replica.

Exercise: Try to define a counter that is decreasing.

Decrease-only Counter

D-Counter

Let n be the number of replicas (nodes of the distributed system).

D-Counter is a CRDT $DC \triangleq \langle \mathbb{Z}^n, \preceq, \curlyvee, \mathbb{Z}, value, decrement_1, \dots, decrement_n \rangle$, where:

- $(x_1, \dots, x_n) \ \preceq \ (y_1, \dots, y_n) \Leftrightarrow \forall 1 \leq i \leq n, x_i \geq y_i$.
- $(x_1, \dots, x_n) \ \curlyvee \ (y_1, \dots, y_n) \triangleq (min(x_1, y_1), \dots, min(x_n, y_n))$.
- $\perp = (0, \dots, 0)$.
- $decrement_k(\langle x_1, \dots, x_n \rangle) \triangleq \langle x_1, \dots, x_k - 1, \dots, x_n \rangle$.
- $value(\langle x_1, \dots, x_n \rangle) \triangleq \sum_{1 \leq i \leq n} x_i$.

Use-cases of G-Counter in Combinatorial Optimization

We explore a search tree in parallel, and wish to get the optimal solution.

- *Branch-and-Bound*: the replicas share a common objective bound, either an increasing or decreasing counter (if maximization or minimization problem).
- *Statistics*: the number of nodes explored in total, number of solutions, number of failed nodes, ...

Positive-Negative Counter (PN-Counter)

Positive-Negative Counter (PN-Counter)

PN-Counter

Let n be the number of replicas (nodes of the distributed system) and $I = \{1, \ldots, n\}$
PN-Counter is a CRDT $PN \triangleq \langle \mathbb{Z}^n \times \mathbb{Z}^n, \ddot{\leq}, \ddot{\sqcup}, \ddot{\bot}, \mathbb{Z}, value, \{increment_i\}_{i \in I}, \{decrement_i\}_{i \in I} \rangle$, where:

- The lattice-theoretic operations are inherited from the Cartesian product.
- The monotone functions are extended as follows:
  - $increment_k(\langle x_1, \ldots, x_n \rangle, D) \triangleq (\langle x_1, \ldots, x_k + 1, \ldots, x_n \rangle, D)$.
  - $decrement_k(G, \langle x_1, \ldots, x_n \rangle) \triangleq (G, \langle x_1, \ldots, x_k + 1, \ldots, x_n \rangle)$.
  - $value(G, D) \triangleq value(G) - value(D)$.

Wouldn't it be possible to do the product of two CRDTs so we don't need to redefine all the operations?

Product of CRDTs

Let $A = \langle L, \leq_A, S_A, value_A, fa_1, \ldots, fa_n \rangle$ and $B = \langle K, \leq_B, S_B, value_B, fb_1, \ldots, fb_n \rangle$.

Product of CRDTs

We have $A \times B$ such that:

- The lattice-theoretic operations are inherited from the Cartesian product.
- Each monotone function $fa_i : A \to A$ and $fb_i : B \to B$ are extended to be applied pairwise on each component:
  - $fa'_i(x, y) \triangleq (fa_i(x), y)$
  - $fb'_i(x, y) \triangleq (x, fb_i(y))$
- $S = S_A \times S_B$ and $value(x, y) = (value_A(x), value_B(x))$.

Does this definition work to obtain PN-Counter?

Product Definition: Positive-Negative Counter (PN-Counter)

The treatment of *value* is not very satisfying and we would prefer to redefine it ourselves, so we can only use the product for combining some operations.

PN-Counter

PN-Counter is a CRDT $PN \triangleq \langle GC \times GC, \mathbb{Z}, \mathit{value} \rangle$ such that:

$$\mathit{value}(x, y) \triangleq \mathit{value}_{GC}(x) - \mathit{value}_{GC}(y)$$

Alternative Definition

PN-Counter is a CRDT $PN' \triangleq \langle GC \times DC, \mathbb{Z}, \mathit{value} \rangle$ such that:

$$\mathit{value}(x, y) \triangleq \mathit{value}_{GC}(x) + \mathit{value}_{DC}(y)$$

Exercise: Find another construction to obtain a similar CRDT (e.g., using lexicographic order).
Exercise: Prove that both definitions are equivalent.

Grow-only Set (G-Set)

Grow-Only Set (G-Set)

Based on another lattice construction: the powerset.

G-Set

Let X be a set of elements. G-Set is a CRDT $GS \triangleq \langle \mathcal{P}(X), \subseteq, \cup, \emptyset, \mathcal{P}(X), \mathit{value}, \mathit{lookup}, \mathit{add} \rangle$, where:

- The lattice-theoretic operations are inherited from the powerset construction.
- $\mathit{value}(S) \triangleq S$.
- $\mathit{lookup}(S, x) \triangleq x \in S$ of type $\mathit{lookup} : \mathcal{P}(X) \times X \to \mathbb{B}$ with $\mathbb{B} = \{\mathit{true}, \mathit{false}\}$.
- $\mathit{add}(S, x) \triangleq S \cup \{x\}$.

Grow-Only Set (G-Set)

Based on another lattice construction: the powerset.

G-Set

Let X be a set of elements. G-Set is a CRDT $GS \triangleq \langle \mathcal{P}(X), \subseteq, \cup, \emptyset, \mathcal{P}(X), \mathit{value}, \mathit{lookup}, \mathit{add} \rangle$, where:

- The lattice-theoretic operations are inherited from the powerset construction.
- $\mathit{value}(S) \triangleq S$.
- $\mathit{lookup}(S, x) \triangleq x \in S$ of type $\mathit{lookup} : \mathcal{P}(X) \times X \rightarrow \mathbb{B}$ with $\mathbb{B} = \{\mathit{true}, \mathit{false}\}$.
- $\mathit{add}(S, x) \triangleq S \cup \{x\}$.

- **Exercise**: Prove that $\mathit{lookup}_x \triangleq \lambda S.\mathit{lookup}(S, x)$ is a monotone function.
- Can we have a "decreasing-only set" CRDT?
- G-Set was the easy case... How can we remove elements?

Decreasing-Only Set (D-Set)

We store what is *not in the set*, instead of what is in the set.

D-Set

Let X be a set of elements. D-Set is a CRDT $DS \triangleq \langle \mathcal{P}(X), \subseteq, \cup, \emptyset, \mathcal{P}(X), \mathit{value}, \mathit{lookup}, \mathit{remove} \rangle$, where:

- The lattice-theoretic operations are inherited from the powerset construction.
- $\mathit{value}(S) \triangleq X \setminus S$.
- $\mathit{lookup}(S, x) \triangleq x \notin S$.
- $\mathit{remove}(S, x) \triangleq S \cup \{x\}$.

Question: Do you foresee any implementation issue?

Two-Phase Set (2P-Set)

Two-Phase Set (2P-Set)

Cartesian product of powerset.

2P-Set

2P-Set is a CRDT $\mathit{TPS} \triangleq \langle \mathit{GS} \times \mathit{GS}, \mathcal{P}(X), \mathit{value}, \mathit{lookup}, \mathit{add}, \mathit{remove} \rangle$, where:

- The lattice-theoretic operations are inherited from the Cartesian product.
- $\mathit{value}(A, R) \triangleq A \setminus R$.
- $\mathit{lookup}((A, R), x) \triangleq x \in A \wedge x \notin R$.
- $\mathit{add}((A, R), x) \triangleq (A \cup \{x\}, R)$.
- $\mathit{remove}((A, R), x) \triangleq (A, R \cup \{x\})$ iff $\mathit{lookup}((A, R), x)$.

We call the set of removed elements R the *tombstone set*.

- **Exercise**: Define this CRDT using the decreasing-only set CRDT defined previously.
- Once we delete an element, can we add it again later?

Observed-Remove Set (OR-Set)

Designing a Set CRDT Supporting Multiple Add/Remove

- Sequentially: the sequence `add(S,x); remove(S,y);` and `remove(S,y);add(S,x);` leads to the same result where $x \in S$ and $y \notin S$ (with $x \neq y$).
- Therefore, we would like our CRDT to have this convergence property as well!
- But some sequences are not commutative, e.g., `add(S,x); remove(S,x);` and `remove(S,x);add(S,x);`

Principle of Permutation Equivalence

Let P be the precondition, Q and Q' the postconditions and $u \| u'$ the concurrent execution.

$$\{P\}u;u'\{Q\} \land \{P\}u';u\{Q'\} \land Q \Leftrightarrow Q' \quad \Rightarrow \quad \{P\}u\|u'\{Q\}$$

What to do when $Q \neq Q'$? In a concurrent execution it would lead to non-determinism.

Designing a Set CRDT Supporting Multiple Add/Remove

- Sequen (S,x);
  leads to
- Therefo
- But sor d
  `remove`

Principle o

Let P be th xecution.
$\{P\}u; u'\{Q$

What to do ism.

Recovering Determinism for `add(S,x) || remove(S,x)`

Possible choices of postconditions:

- $\{\bot \in S\}$ (error mark)
- $\{x \in S\}$ (add wins—next slide)
- $\{x \notin S\}$ (remove wins)
- $\{add(S,x) >_{CLK} remove(S,x) \Leftrightarrow x \in S\}$ (last writer wins (LWW))

Observed-Remove Set (OR-Set)

Intuitions

- Given n replicas, we assign a unique ID to each of them.
- We count the number of local operations $k \in \mathbb{N}$ performed on the set.
- Each time we add or remove an element in the set, we stick the unique pair $(id, k) \in \mathbb{N}^2$ to the element.
- Let **UID** $\triangleq \mathbb{N} \times \mathbb{N}$ be the set of all unique identifiers.

Exercise: Define the corresponding CRDT.

Observed-Remove Set (OR-Set)

Let $gen_i(A, T) \triangleq (i, 1 + \max\{k \in \mathbb{Z} \mid \exists x,\ ((i,k), x) \in (A \cup T)\})$.

OR-Set

OR-Set is a CRDT $ORS \triangleq \langle \mathcal{P}(\mathbf{UID} \times X)^2, \leq, \sqcup, (\emptyset, \emptyset), \mathcal{P}(X), value, lookup, add, remove \rangle$:

- $(A_1, T_1) \leq (A_2, T_2) \Leftrightarrow (A_1 \cup T_1) \subseteq (A_2 \cup T_2) \wedge T_1 \subseteq T_2$.
- $(A_1, T_1) \sqcup (A_2, T_2) \triangleq ((A_1 \setminus T_2) \cup (A_2 \setminus T_1), T_1 \cup T_2)$.
- $value(A, T) \triangleq \{x \in X \mid \exists uid,\ (uid, x) \in A\}$.
- $lookup((A, T), x) \triangleq x \in value(A, T)$.
- $add_i((A, T), x) \triangleq (A \cup \{(gen_i(A, T), x)\}, T)$.
- $remove_i((A, T), x) \triangleq \textbf{let } R = \{(uid, x) \mid (uid, x) \in A\} \textbf{ in } (A \setminus R, T \cup R)$.

Exercise: Prove the order $\leq$ and the join $\sqcup$ are consistent, i.e. $X \leq Y \Leftrightarrow X \sqcup Y = Y$.
Exercise: Find a way to define this CRDT without having to redefine yourself the lattice operations.

Operation-based CRDTs (Formally)

Causal Order

Definition

Causal order is a partial order $\prec$ on messages such that $m_1 \prec m_2$ if the replica that sent m_2 did so after receiving m_1, or if the same replica sent m_1 before m_2.

Replicas that receive messages in causal order means that a replica should not receive a message m_2 until after it has received all messages $m_1 \prec m_2$.

Definition of Operation-based CRDT

An operation-based CRDT is a tuple $\langle \Sigma, \sigma^0, \mathit{eval}, \mathit{prepare}, \mathit{effect} \rangle$ where:

- Σ is a set of states.
- $\sigma^0 \in \Sigma$ is the initial state.
- $\mathit{eval}(q, \sigma)$: read-only evaluation of the query q on state σ.
- $\mathit{prepare}(o, \sigma, r)$: prepares a message m given an operation o by replica r in state σ.
- $\mathit{effect}(m, \sigma)$: applies a message m on state σ, and returns the result. When convenient, we write this function as $m \cdot \sigma$.

Further, to ensure concurrent operations commute, we require that:

$$m_1 \cdot (m_2 \cdot \sigma) = m_2 \cdot (m_1 \cdot \sigma)$$

Because operations are not, in general, idempotent, it is essential that an exactly-once messaging mechanism is used.

Operation-based G-Counter

G-Counter

Let $\langle \mathbb{Z}, 0, \mathit{eval}, \mathit{prepare}, \mathit{effect} \rangle$ where:

- $\mathit{eval}(\texttt{value}, \sigma) = \sigma$.
- $\mathit{prepare}(\texttt{add}(n), \sigma, r) = \texttt{add}(n)$.
- $\mathit{effect}(\texttt{add}(n), \sigma) = n + \sigma$.

Exercise: Define an operation-based grow-only set CRDT.

Operation-based G-Set

G-Set

Let $\langle \mathcal{P}(X), \emptyset, \mathit{eval}, \mathit{prepare}, \mathit{effect} \rangle$ where:

- $\mathit{eval}(\mathtt{value}, \sigma) = \sigma$.
- $\mathit{eval}(\mathtt{contains}, x, \sigma) = x \in \sigma$.
- $\mathit{prepare}(\mathtt{add}(x), \sigma, r) = \mathtt{add}(x)$.
- $\mathit{effect}(\mathtt{add}(x), \sigma) = \{x\} \cup \sigma$.

Distributed Algorithm for Operation-based CRDT

On each replica r, we have the following event-based algorithm:

```
**state** $\sigma \in \Sigma$, initially $\sigma^0$
**on** operation($o$) :
    $m \leftarrow \texttt{prepare}(o, \sigma, r)$
    $\sigma \leftarrow \texttt{effect}(m, \sigma)$
    Broadcast $m$ to other replicas
**on** receive($m$) :
    $\sigma \leftarrow \texttt{effect}(m, \sigma)$
**on** query($q$) :
    **return** $\texttt{eval}(q, \sigma)$
```

Note: Messages are assumed to be received in causal order.

References

- **General Survey**: P. S. Almeida, *Approaches to Conflict-free Replicated Data Types*, ACM Computing Survey, Sep. 2024.
- On composition of CRDTs:
    - **Operation-based**: M. Weidner, H. Miller, and C. Meiklejohn, *Composing and decomposing op-based CRDTs with semidirect products*, POPL 2020.
    - **State-based**: C. Baquero, P. S. Almeida, A. Cunha, and C. Ferreira, *Composition in State-based Replicated Data Types*, Bulletin of EATCS 3, no. 123 (2017).