Formally Verified Semilattice Class for Conflict-free Replicated Data Types (CRDTs)

Madhav Ram - PROOF101 - April 2026

Semilattice

Semilattice - partially ordered set that has a join function (a least upper bound) for any nonempty finite subset.

Building block of CRDTs - defined as a type class

join() function must follow:

- **Associativity** - ∀ a b c, join (join a b) c = join a (join b c)
- **Commutativity** - ∀ a b, join a b = join b a
- **Idempotency** - ∀ a, join a a = a

Partial Order

join() on an ancestor and a newer state must ensure the newer state is up to date on everything about the ancestor's state.

We can track this using a <= operator!

Theorems on semilattice_le operator:

- **Reflective** - (a : α) : semilattice_le a a - follows from join's idempotency
- **Antisymmetric** - (a b : α) : semilattice_le a b → semilattice_le b a → a = b - join's commutativity
- **Transitive -** follows from join's associativity

  (a b c : α) : semilattice_le a b → semilattice_le b c → semilattice_le a c

Inflationary - Monotonic nature of updates

update() function of a CRDT must ensure the values only increment its state.

Decrements are support via tombstone sets or additional CRDT that hold decrement steps.

Theorems for all CRDTs proves updates are inflationary

CRDTs - G Counter, Vector Clock, PN Counter,

G-Counter: A counter that only increments, using max to safely merge states across a network. *Gcounter := Nat*

PN-Counter: A pair of G-Counters (one tracking additions, one tracking subtractions) whose difference is the actual value. *PNcounter := Gcounter × Gcounter*

Vector Clock: A map that tracks the logical time of every node in a network, merging by taking the maximum value for each individual node.

VectorClock := NodeId → Nat

```
-- Initial State of a Semilattice used for many CRDTs
class BoundedSemilattice (α : Type) extends Semilattice α where
  bottom : α
  join_bottom : ∀ x, Semilattice.join x bottom = x

-- (Update) Functions on Semilattice are "Grow-only" - monotonic
def is_inflationary {α : Type} [Semilattice α] (f : α → α) : Prop :=
  ∀ x, semilattice_le (f x) x

-- G Counter Implementation on BoundedSemilattice
abbrev Gcounter := Nat  -- def does not inherit properties of original type and is unable to
instance : BoundedSemilattice Gcounter where
  join := Nat.max

  join_asoc := Nat.max_assoc
  join_comm := Nat.max_comm
  join_idem := Nat.max_self

  bottom := 0
  join_bottom := Nat.max_zero

def gcounter_increment (n: Gcounter) : Gcounter :=
 n + 1

theorem gcounter_increment_is_inflationary : is_inflationary gcounter_increment := by
  unfold is_inflationary
  unfold semilattice_le
  unfold gcounter_increment
  intro x
  apply Nat.max_eq_left
  apply Nat.le_succ
```

G-Counter

Tactics: funext and ext

funext (Function Extensionality): Proves that two functions are identical by checking all inputs (i) and proving they produce the exact same output.

ext (Extensionality): Proves that two structures are identical by splitting the proof into separate, smaller goals for each of their individual component.

```
instance : BoundedSemilattice PNcounter where
  join := pnc_join
  bottom := pnc_bottom

  join_asoc a b c := by
    -- ext breaks down into cases for my PNcounter pair
    -- this allows to match type with Nat.max lemmas
    ext
    · exact Nat.max_assoc a.1 b.1 c.1
    · exact Nat.max_assoc a.2 b.2 c.2

  join_comm a b := by
    ext
    . exact Nat.max_comm a.1 b.1
    . exact Nat.max_comm a.2 b.2

  join_idem a := by
    ext
    . exact Nat.max_self a.1
    . exact Nat.max_self a.2

  join_bottom a := by
    ext
    . exact Nat.max_zero a.1
    . exact Nat.max_zero a.2
```

Showcase of ext in PN-Counter

References

https://jhellerstein.github.io/blog/crdt-intro/ and other blogs in the series.

https://avichalp.me/posts/2023-05-07-crdts-in-a-nutshell/

https://en.wikipedia.org/wiki/Semilattice

https://hydro.run/ - A Rust framework for correct and performant distributed systems

Link to git codebase

https://github.com/madhav6ram/fv-semilattice/tree/main