


# Modular Verification of State-Based CRDTs in Separation Logic (Artifact)

Abel Nieto 

Aarhus University, Denmark

Arnaud Daby-Seesaram 

ENS Paris-Saclay, France

Léon Gondelman 

Aarhus University, Denmark

Amin Timany 

Aarhus University, Denmark

Lars Birkedal 

Aarhus University, Denmark

## Abstract

This is the documentation of the artifact for the paper “Modular Verification of State-Based CRDTs in Separation Logic”. The artifact consists of a Coq

formalization of the safety proofs for state-based CRDTs described in the paper. The Coq proofs are written in the Aneris distributed separation logic.

**2012 ACM Subject Classification** Theory of computation → Program verification; Theory of computation → Distributed algorithms; Theory of computation → Separation logic

**Keywords and phrases** separation logic, distributed systems, CRDT, replicated data type, formal verification

**Digital Object Identifier** 10.4230/DARTS.9.2.15

**Funding** This work was supported in part by a Villum Investigator grant (no. 25804), Center for Basic Research in Program Verification (CPV), from the VILLUM Foundation.

**Related Article** Abel Nieto, Arnaud Daby-Seesaram, Léon Gondelman, Amin Timany, and Lars Birkedal, “Modular Verification of State-Based CRDTs in Separation Logic”, in 37th European Conference on Object-Oriented Programming (ECOOP 2023), LIPIcs, Vol. 263, pp. 22:1–22:27, 2023.

<https://doi.org/10.4230/LIPIcs.ECOOP.2023.22>

**Related Conference** 37th European Conference on Object-Oriented Programming (ECOOP 2023), July 17–21, 2023, Seattle, Washington, United States

**Evaluation Policy** The artifact has been evaluated as described in the ECOOP 2023 Call for Artifacts and the ACM Artifact Review and Badging Policy.

## 1 Scope

Below we reproduce the paper’s list of contributions and, for each contribution, list how the proof development backs it up.

### 1. Claimed contribution:

We give the first modular specification of a general class of state-based CRDTs. Our specifications are given in the Aneris distributed separation logic and our proofs are mechanized in Coq.

By a “general class of state-based CRDTs” we mean the StLib library. The library has a modular architecture and as such can be instantiated as detailed in the paper to generate multiple verified example CRDTs.



© Abel Nieto, Arnaud Daby-Seesaram, Léon Gondelman, Amin Timany, and Lars Birkedal;  
licensed under Creative Commons License CC-BY 4.0

Dagstuhl Artifacts Series, Vol. 9, Issue 2, Artifact No. 15, pp. 15:1–15:5



DAGSTUHL  
ARTIFACTS SERIES

Dagstuhl Artifacts Series  
Schloss Dagstuhl – Leibniz-Zentrum für Informatik,  
Dagstuhl Publishing, Germany



## 15:2 Modular Verification of State-Based CRDTs in Separation Logic (Artifact)

The directory structure in Section 2 shows the locations of each example CRDT.

For each example we include the example's code as well as a proof that the library can be instantiated with the example (this involves defining the relevant lattice and proving that the implementation meets the specifications for the merge, mutator, and init functions).

e.g. for the Grow Only Set:

- `examples/grow_only_set/grow_only_set_code.v` contains the implementation
- `examples/grow_only_set/grow_only_set_proof.v` contains the proof

At the end of the file the lemma `gos_init_spec` shows that `statelib` can be instantiated with our grow-only set implementation.

### 2. Claimed contribution:

Furthermore, when taken together with Nieto et al. [1], our work provides a unified framework for the specification and verification of both kinds of CRDTs. This is because our specifications of state-based CRDTs are compatible with Nieto et al.'s specifications of op-based CRDTs. We emphasize this point by re-verifying the example client program in Nieto et al. that uses a positive-negative counter CRDT, except we swap their opbased counter by a state-based equivalent. Crucially, the client's safety proof remains virtually unchanged, showing that it is possible to specify CRDTs while hiding their implementation strategy.

The CRDT client can be found under `examples/pncounter/pncounter_use_case.v` and is adapted from Nieto et al. [1]. What is going on here:

- The previous paper on operation-based CRDTs proved that a client program can use and reason about an op-based PN counter. Their proof can be found under `oplib/examples/pncounter_use_case/pncounter_use_case.v`
- We also implemented a PN counter, this time using state-based techniques in `statelib/examples/pncounter/pncounter_use_case.v`
- We copied their proof (and client code) and modified it to work for our state-based implementation.

We had to make just a few technical changes to the proof so that it would go through.

The reader can look at the differences between the two proofs by doing

```
diff pncounter_use_case.v
    ../../../../oplib/examples/pncounter_use_case/pncounter_use_case.v"
```

from the `examples/pncounter` directory.

Most of the changes in the diff are either importing different files, or using one library vs the other. With additional proof engineering effort, even those changes could be eliminated.

By "our specifications are compatible with Nieto et al.'s specifications for op-based CRDTs" we mean this:

- The file `examples/crdt/spec` contains in `crdt_resources.v` the `Class CRDT_Res_Mixin` interface with lemmas on the use of CRDT resources (these are the resources that a client would work with when reasoning about a CRDT).
- Notice that the directory structure suggests this resource interface is shared between the `OpLib` and `StateLib` developments.
- One can check this is the case by grepping for instances of the typeclass above.

```
grep -r CRDT_Res_Mixin | grep Instance
oplib/proof/resources.v:
Global Instance oplib_res : CRDT_Res_Mixin Mdl S LogOp := {
statelib/examples/pncounter/pncounter_proof.v:
Global Program Instance pn_CRDT_Res : CRDT_Res_Mixin M S CtrOp :=
statelib/resources/resources.v:
Global Instance StLib_CRDT_Res_Mixin: CRDT_Res_Mixin _ _ CRDT_Op :=
```

Notice there's one instance in `OpLib` and two in `StateLib`. The additional instance `statelib/examples/pncounter/pncounter_proof.v` has to do with the fact that our implementation of a PN counter is not obtained directly as an instantiation of `StateLib` (in which case a new instance wouldn't be needed), but instead uses some wrapper code.

The important point is that both libraries (`OpLib` and `StateLib`) are able to instantiate the *same* resource interface.

### 3. Claimed contribution:

We give the first formal proof that state-based CRDTs are causally-consistent.

To improve the presentation, in the paper we separated the causality-related lemmas into two. First, lemma 4 shows causality at the model level (in terms of state-transition systems). Later, lemma 5 shows causality using separation logic resources.

In the Coq formalization, both proofs are handled together as part of the lemma `StLib_OwnLocalStat_GlobSnap_Causality` in file `statelib/resources/resources.v`.

Additionally, later on in that same file we instantiate the resource interface for state-based CRDTs:

```
Global Instance StLib_CRDT_Res_Mixin: CRDT_Res_Mixin _ _ CRDT_Op :=
```

As part of that typeclass instance, we need to provide a proof that state-based resources are causally-consistent:

```
LocState_GlobSnap_Causality := StLib_OwnLocalState_GlobSnap_Causality;
```

The above lemma (`LocState_GlobSnap_Causality`) together with the fact that the specifications for `StateLib`'s `get` and `update` use the local state resource, show that state-based CRDTs are causally-consistent.

### 4. Claimed contribution:

We evaluate our approach by verifying a set of example CRDTs from the literature. The evaluation shows that our techniques can handle a variety of CRDTs, including counters, sets, and higher-order combinators.

The example CRDTs can be found under `statelib/examples`.

## 2 Content

The top-view of the structure of the development is as follows:

```
.
| - aneris
  (* most relevant folders for this project *)
  | | - examples
  | | | - rcb           %% Reliable causal broadcast (used for crdt oplib)
  | | | - crdt         %% Conflict replicated data types (both stlib and oplib)
  | | - aneris_lang    %% Aneris program logic on top of which examples are built
  (* other folders *)
  | - trillium         %% refinement program logic
  | - external         %% dependencies (e.g. Iris)
  | | - lib            %% some logical theories
  | | - prelude        %% idem.
  | | - algebra        %% idem.
```

## 15:4 Modular Verification of State-Based CRDTs in Separation Logic (Artifact)

The most relevant folder related to the material presented in the paper is `stlib_crdt_artifact/aneris/aneris/examples/crdt/` with the following correspondence with the paper sections:

```
| - spec                %% Common logical interface to reason about CRDTs
                        %% in separation logic (Sect.4)
| - statelib           %% Statelib CRDT formal development
| | - user_model       %% mathematical model of stlib crdts,
                        %% e.g., model with lub, mutator, and lattice (Sect. 7.1)
| | - proof            %% Contains the implementation (Sect. 5), the spec (Sect. 6),
                        %% and the proof (Sect. 7) of stlib.
| | - STS              %% State transition system (Sect. 7.1)
| | - resources        %% Instantiation of abstract CRDT resources for
                        %% stlib (Sect. 7.2)
| | - time             %% Instantiation of abstract notion of time for stlib
| | - examples         %% Contains the implementation and the proof of
                        %% the examples (Sect.8)
| | | - prod_comb
| | | - gcounter
| | | - grow_only_set
| | | - map_comb
| | | - pncounter
| - oplib              %% Oplib CRDT formal development (relevant in particular
                        %% for footnote 9 on p. 23)
```

### Correspondence between figures in the paper and Coq development

Paper	Coq development
Figure 1: Op-based and state-based OCaml implementations of a grow-only counter	<code>/aneris/aneris/examples/crdt/statelib/examples/gcounter/*.v</code>
Figure 3: CRDT resources and selected lemmas	<code>/aneris/aneris/examples/crdt/spec/crdt_resources.v</code>
Figure 5: StateLib implementation and a G-Set example	<code>/aneris/aneris/examples/crdt/statelib/statelib_code.v</code> <code>/aneris/aneris/examples/crdt/statelib/examples/grow_only_set/grow_only_set_code.v</code>
Figure 6: Internal specifications and Figure 7: External specifications	<code>aneris/aneris/examples/crdt/statelib/proof/spec.v</code>
Lemmas 4 and 5 (Causality)	Lemma <code>StLib_OwnLocalState_GlobSnap_Causality</code> in file <code>statelib/resources/resources.</code>

### 3 Getting the artifact

The artifact endorsed by the Artifact Evaluation Committee is available free of charge on the Dagstuhl Research Online Publication Server (DROPS). A VirtualBox VM with Coq installed and our development can be obtained from <https://zenodo.org/record/7719423>.

The password for the default VM user `vboxuser` is `changeme`.

### 4 Tested platforms

The VM should run anywhere that supports VirtualBox 7.0.

## 5 License

The artifact is available under a Creative Commons Attribution 4.0 International license.

## 6 MD5 sum of the artifact

806cc18ae49047142453ecbc47175c8e

## 7 Size of the artifact

5.8 GiB

---

## References

- 1 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):1788–1816, 2022.