Technical Report

Cheng Li[♭], Nuno Preguiça[‡], Rodrigo Rodrigues[*]
[♭]*University of Science and Technology of China,* [‡]*NOVA LINCS & FCT, Univ. NOVA de Lisboa,*
[*]*INESC-ID & IST, Universidade de Lisboa*

1 Preliminaries

1.1 System model

We assume a geo-distributed system with state fully replicated across k sites denoted by $site_0 \ldots site_{k-1}$, where each site hosts a replica, and each replica runs as a deterministic state machine. In the rest of the document, the terms "site" and "replica" are interchangeable.

The system defines a set of operations $\mathscr{U}$ manipulating a set of reachable states $\mathscr{S}$. Each operation u is initially submitted by a user at one site which we call u's *primary site* and denote $site(u)$. An operation is defined mathematically as a function that receives the current state of the system and returns another function corresponding to its side effects. We refer to the former function as the *generator* function, denoted by g_u; this generator function, when applied to a given state $S \in \mathscr{S}$, returns a *shadow* function or *shadow operation*, denoted $h_u(S)$.

Implementation-wise, the generator function will first execute in a *sandbox* against the current state of the replica at the user's primary site, without interference from other concurrent operations. In this phase, the execution only identifies what changes u would introduce to state S that is observed by u and will not commit these changes. At the end of executing h_u, the identified side-effect or shadow operation $h_u(S)$ will be sent and applied across all replicas including the primary site.

A desirable property is that all replicas that have applied the same set of shadow operations are in the same state, i.e., the underlying system offers **state convergence**. In addition, the system maintains a set of application-specific **invariants**. For instance, an online shopping service cannot sell more items than those available in stock. To capture this notion, we define the function $valid(S)$ to be *true* if state S satisfies all these invariants and *false* otherwise.

1.2 RedBlue consistency

Our prior work RedBlue consistency [2] is based on an explicit division of shadow operations into blue operations whose order of execution can vary from site to site, and red operations that must execute in the same relative order at all sites. For guiding developers in making use of RedBlue consistency, this work identified that a sufficient condition for ensuring state convergence is that a shadow operation must be labeled red if it is not globally commutative. For ensuring that invariants are maintained, a second sufficient condition was identified, stating the following: the replicas will only transition between valid states if we label as red all shadow operations that may violate an invariant when applied against a different state from the one they were generated from. For the remaining shadow operations, which have passed the two condition checks, we can safely label them blue.

```
boolean placeBid(int
      itemId, int clientId,
      int bid){
 boolean result = false;
 beginTxn();
 if(open(itemId)){
   createShadowOp(placeBid
         ', itemId,
         clientId, bid);
   result = true;
 }
 commitTxn();
 return result;
}
```

(a) Original placeBid operation.

```
int closeAuction(int itemId){
 int winner = -1;
 beginTxn();
 close(itemId);
 winner = exec(SELECT userId
       FROM bidTable WHERE
       iId = itemId ORDER BY
       bid DESC limit 1);
 createShadowOp(closeAuction
       ',itemId, winner);
 commitTxn();
 return winner;
}
```

(c) Original closeAuction operation.

```
placeBid'(int itemId, int
      clientId, int bid){
  exec(INSERT INTO
       bidTable VALUES (
       bid, clientId, itemId
       ));
}
```

(b) Shadow placeBid' operation.

```
closeAuction'(int itemId, int
       winner){
 close(itemId);
 exec(INSERT INTO
       winnerTable VALUES (
       itemId, winner));
}
```

(d) Shadow closeAuction' operation.

Figure 1: Pseudocode for the `placeBid` and `closeAuction` operations of an auction site

2 Partial Order-Restriction Consistency

2.1 Motivating example

We illustrate the limitations of coarse-grained labeling schemes like RedBlue consistency through an ebay-like auction service in Fig. 1, where an operation `placeBid` (Fig. 1(a)) creates a new bid for an item if the corresponding auction is still open, and an operation `closeAuction` (Fig. 1(c)) closes an auction and declares a single winner. In this example, the application-specific invariant is that the winner must be associated with the highest bid across all accepted bids. The other two subfigures (Fig. 1(b) and Fig. 1(d)) depict the shadow operations of the two prior operations, respectively, guaran-

teeing that these shadow operations apply changes in a commutative fashion regardless of execution order.

When applying RedBlue consistency to replicate such an auction service, we note that the concurrent execution under weak consistency of a `placeBid` with a bid higher than all accepted bids and a `closeAuction` can lead to the violation of the application invariant. This happens because the generation of `closeAuction'` will ignore the highest bid created by the concurrent shadow `placeBid'`. Unfortunately, the only way to address this issue in RedBlue consistency is to label both shadow operations as strongly consistent, i.e., all shadow operations of either type will be totally ordered w.r.t each other, which will incur in a high overhead in geo-distributed settings. Intuitively, however, there is no need to order pairs of `placeBid'` shadow operations, since a bid coming before or after another does not affect the winner selection. This highlights that a coarse-grained operation classification into two levels of consistency can be conservative, and some services could benefit from additional flexibility in terms of the level of coordination.

To overcome these limitations of RedBlue consistency, we next propose Partial Order-Restrictions consistency (or short, PoR consistency), a novel consistency model that allows the developer to reason about various fine-grained consistency requirements in a single system. The key intuition behind our proposal is that this model is generic and can be perceived as a set of restrictions imposed over admissible partial orders across the operations of a replicated system.

2.2 Defining PoR consistency

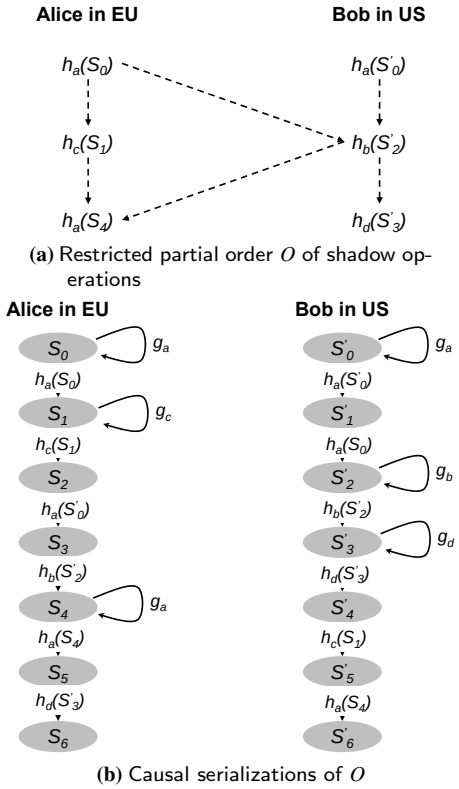
The definition of PoR consistency includes three important components: (1) a set of restrictions, which specify the visibility relations between pairs of operations; (2) a restricted partial order (or short, R-order), which establishes a (global) partial order of operations respecting operation visibility relations; and (3) a set of site-specific causal serializations, which correspond to total orders in which the operations are locally applied. We define these components formally as follows:

Definition 1 (Restriction)**.** *Given a set of operations U, a* restriction *is a symmetric binary relation on $U \times U$.*

For any two operations u and v in U, if there exists a restriction relation between them, we denote this relation as $r(u,v)$.

Definition 2 (Restricted partial order)**.** *Given a set of operations U, and a set of restrictions R over U, a* restricted partial order *(or short,* R-order*) is a partial order $O = (U, \prec)$ with the following constraint: $\forall u, v \in U$, $r(u,v) \in R \implies u \prec v \vee v \prec u$.*

We also say that the restrictions in R are met in the corresponding R-order if this order satisfies the above definition. This definition places constraints on a global view of a replicated system; however, it fails to explain how each individual replica at every site will behave according to this global view. To introduce these local views, we have to recall the system model our consistency definition is built on. When user requests are accepted by any site, that site executes their generator operations and creates corresponding shadow operations which will be replicated across all sites. In addition, every site not only commits shadow operations created by itself, but also applies remote ones shipped from all other sites against its local state. For a site i, we denote V_i as its generator operation set. The following definition models the execution of each site as a growing linear extension of the global R-order, which incorporates a notion of causality, due to the fact that the visibility dependencies that are established when operations are initially generated, are then preserved while the corresponding shadow operations are replicated.



(a) Restricted partial order O of shadow operations

(b) Causal serializations of O

Figure 2: Restricted partial order and causal legal serializations for a system spanning two sites. There exists a restriction $r(h_a(S), h_b(S))$ for all valid S. Dotted arrows in Fig. 2a indicate dependencies between shadow operations. Loops in Fig. 2b represent generator operations.

Definition 3 (Causal legal serialization)**.** *Given a site i, an R-order $O = (U, \prec)$ and the set of generator operations V_i received at site i, we say that $O_i = (U \cup V_i, <_i)$ is an* i-causal legal serialization *(or short, a* causal seri-

alization*) of O if*

- *O_i is a total order;*
- *$(U, <_i)$ is a legal serialization of O;*
- *For any $h_v(S) \in U$ generated by $g_v \in V_i$, S is the state obtained after applying the sequence of shadow operations preceding h_v in O_i;*
- *For any $g_v \in V_i$ and $h_u(S) \in U$, $h_u(S) <_i g_v$ in O_i iff $h_u(S) \prec h_v(S')$ in O.*

Definition 4 (Partial Order-Restrictions consistency)**.** *A replicated system $\mathscr{S}$ spanning k sites with a set of restrictions R is* Partial Order-Restrictions consistent *(or short,* PoR consistent*) if each site i applies shadow operations according to an i-causal serialization of R-order O.*

Fig. 2 shows a restricted partial order and its causal legal serializations executed at two sites, namely EU and US. Since we restrict pairs of shadow operations where one corresponds to a and the other to b, when the US site executes a generator of b, g_b, it realizes that the shadow operation it would generate may need to be restricted w.r.t a concurrent shadow operation initially triggered at the EU site. As a result, g_b at the US site must wait until the respective concurrent shadow operation $h_a(S_0)$ gets propagated from the EU site to Bob's site. Then g_b will read the state introduced by locally applying $h_a(S_0)$ from Alice, and produce a shadow operation $h_b(S_2')$. Note that this production will establish a dependency between $h_a(S_0)$ and $h_b(S_2')$ (as shown in Fig. 2a), thus enforcing that they cannot be applied in different relative orders in all causal legal serializations (as shown in Fig. 2b). Unlike these two shadow operations, we do not restrict any pair of shadow operations of a; as such, the first operations issued by both Alice and Bob will be concurrently executed without being aware of each other. This example indicates the flexibility and performance benefits of having PoR consistency, compared with RedBlue consistency, since under the latter model all shadow operations of a and b would be serialized w.r.t each other.

3 Restriction inference

When replicating a service under PoR consistency, the first step is to infer restrictions to ensure two important system properties, namely state convergence and invariant preservation. The major challenge we face is to identify a small set of restrictions for making the replicated service eventually converge and never violate invariants so that the amount of required coordination is minimal. With regard to state convergence, we take a similar methodology adopted in prior research [3, 2, 1], which is to check operation commutativity. However, unlike RedBlue consistency, under which all operations that are not globally commutative must be totally ordered, PoR consistency only requires that an operation must be ordered w.r.t another one if they do not commute.

To always preserve application-specific invariants, instead of totally ordering all non-invariant safe shadow operations, i.e., those that potentially transition from a valid state to an invalid one, we try to isolate the operations that exclusively contribute to an invariant violation from the rest. To do so, we introduce a new concept, called an `I-conflict set`, which defines a minimal set of shadow operations that lead to an invariant violation when they are running concurrently in a coordination-free manner. By minimal, we mean that by removing any shadow operation from such a set, the violation will no longer persist. To identify a minimal set of restrictions, we first perform an analysis over any subsets of the shadow operation set to discover all `I-conflict` sets. Then, for any such set, adding a restriction between any pair of its operations is sufficient to eliminate the problematic executions.

3.1 State convergence

A PoR consistent replicated system is state convergent if all its replicas reach the same final state when the system becomes quiescent, i.e., for any pair of causal legal serializations of any R-order, L_1 and L_2, we have $S_0(L_1) = S_0(L_2)$, where S_0 is a valid initial state. We state a necessary and sufficient condition to achieve this in the following theorem.

Theorem 5. *A PoR consistent system $\mathscr{S}$ with a set of restrictions R is* ***convergent****, if and only if, for any pair of its shadow operations u and v, $r(u,v) \in R$ if u and v don't commute.*

In order to prove the above theorem, we introduce the following three lemmas along with their proofs. As the generator operations never change state, to simplify the proofs, we remove them from the formalization.

The first lemma asserts that, given a causal legal serialization, swapping two adjacent operations in the causal legal serialization that are not ordered by the underlying R-order results in another causal legal serialization.

Lemma 6. *Given a causal legal serialization $O_i = (U, <_i)$ of R order $O = (U, \prec)$ with operations $u, v \in U$ such that $u <_i v$ and $u \not\prec v$ and there exists no s such that $u <_i s <_i v$, and let $P = \{p | p \in U \wedge p <_i u\}$ and $Q = \{q | q \in U \wedge v <_i q\}$. The serialization $O_k = (U, <_k)$ where*

- *$\forall p, q \in P \cup Q : p <_k q \iff p <_i q$,*
- *$\forall p \in P : p <_k v$,*
- *$v <_k u$,*
- *$\forall q \in Q : u <_k q$*

is a legal serialization.

Proof: It suffices to show that $\forall r, s \in U : r <_k s$ is compatible with $\prec$. To do so, we consider the following six cases:

- **Case 1:** $r,s \in P \cup Q$. Since O_i is a legal serialization, each $r <_i s$ is compatible with $\prec$ by definition. By construction $\forall p,q \in P \cup Q : r <_k s \iff r <_i s$, so each $r <_k s$ is also compatible with $\prec$.
- **Case 2:** $r \in P$, $s = v$. $r <_k s$ is compatible with $\prec$ by similar logic as above.
- **Case 3:** $r = u$, $s \in Q$. $r <_k s$ is compatible with $\prec$ by similar logic as above.
- **Case 4:** $v <_k u$. Since $u \not\prec v$, $v <_k u$ is compatible with $\prec$.
- **Case 5:** $r \in P$, $s = u$. Since $v <_k u \wedge \forall p \in P : p <_k v \implies p <_k u$. By the construction of P, $\forall p \in P : p <_k u \iff p <_i u$. So each $r <_k s$ is also compatible with $\prec$.
- **Case 6:** $r = v$, $s \in Q$. Since $v <_k u \wedge \forall q \in Q : v <_k q \implies v <_k q$. $r <_k s$ is compatible with $\prec$ by similar logic as above.

As $U = P \cup Q \cup \{u,v\}$, by all above cases, $\forall r,s \in U : r <_k s$ is compatible with $\prec$. □

Lemmas 7 asserts that given an R-order and one of its causal legal serializations, if there exists a pair of shadow operations u and v that is not ordered by the R-order, then there exists an adjacent pair of shadow operations between u and v in that serialization that are not ordered by the R-order.

Lemma 7. *Given a legal serialization $O_i = (U, <_i)$ of R order $O = (U, \prec)$, if $\exists u,v \in U$ such that $u <_i v$ and $u \not\prec v$, let $U' = \{u,v\} \cup \{q | u <_i q \wedge q <_i v\}$, then $\exists r,s \in U'$ such that $r <_i s \wedge r \not\prec s \wedge \nexists p \in U' : r <_i p \wedge p <_i s$.*

Proof: We prove this by performing the following exhaustive analysis. The analysis terminates when the required pair of elements is found.

Let's start with u,v. Consider Q to be the sequence of elements strictly between u and v, i.e., $Q = \{q \in U | u <_i q \wedge q <_i v\}$. There are two cases we have to analyze:

- **Case 1:** Q is empty. This implies that u and v are adjacent, so the analysis terminates.
- **Case 2:** Q is not empty. This implies that u and v are not adjacent. Consider p to be the first element in Q according to $<_i$, i.e., $p \in Q : \forall q \in Q \setminus \{p\}, p <_i q$. There are two cases to consider:
  - **Case 2a:** $u \not\prec p$. It follows that p is the successor of u in O_i, then u,p is the adjacent pair that is not ordered by O. The analysis terminates.
  - **Case 2b:** $u \prec p$. It follows from the assertion that $u \not\prec v$ and the transitivity of $\prec$ that $p \not\prec v$. Then we run the analysis from the beginning with p,v. Since we are removing the first element of the sequence Q, the analysis will either eventually terminate with an empty sequence, or before that.

□

The third lemma h asserts that two causal legal serializations of an R-order that differ in the order of exactly one pair of adjacent shadow operations are state convergent, if the two operations commute.

Lemma 8. *Assume $O_i = (U, <_i)$ and $O_j = (U, <_j)$ are both legal serializations of R-Order $O = (U, \prec)$ that are identical except for two adjacent operations u and v such that $u <_i v$ and $v <_j u$ and that u and v commute. Then $S_0(O_i) = S_0(O_j)$.*

Proof: Let P and Q be the greatest common prefix and suffix of O_i and O_j, respectively. Further, let $S_P = S_0(P)$, $S_{uv} = S_P + u + v$, and $S_{vu} = S_P + v + u$. By the definition of operation commutativity, $S_{uv} = S_{vu}$. It then follows from the definition of a deterministic state machine that $S_{uv}(Q) = S_{vu}(Q)$. By a similar argument, the final state reached by sequentially executing operations in O_i against S_0 according to $<_i$ is equal to the final state obtained by sequentially applying operations in Q against S_{uv} according to $<_i$, namely $S_0(O_i) = S_{uv}(Q)$. By a similar argument, we know $S_0(O_j) = S_{vu}(Q)$. Finally, we have $S_0(O_i) = S_0(O_j)$. □

After adapting these lemmas from Chapter **??** to PoR consistency, we use them to construct the proof of the state convergence theorem (Theorem 5) as follows:

Proof: ($\Leftarrow$:) We first show that if for any pair of non-commuting operations of $\mathscr{S}$, a restriction between this pair of operations is in R, then the PoR consistent system $\mathscr{S}$ is convergent. To prove this, it is sufficient to show that any pair of legal serializations of their underlying R-order O, O_i and O_j, is state convergent, i.e., $S_0(O_i) = S_0(O_j)$. There are two cases to consider:

Case 1: $O_i = O_j$. The underlying deterministic state machine ensures that $S_0(O_i) = S_0(O_j)$.

Case 2: $O_i \neq O_j$, in which case $\exists u,v \in U$ such that $u <_i v$ and $v <_j u$. Since both O_i and O_j are legal serializations of O, it follows that $u \not\prec v$ and $v \not\prec u$. It then follows from Lemma 7 that we can find an adjacent pair of operations r,s in both O_i and O_j such that $r <_i s \wedge s <_j r \wedge r \not\prec s \wedge s \not\prec r$. We construct a new serialization O_{i+1} by duplicating O_i but swapping the order of r and s in O_{i+1}, i.e., Q_i and Q_{i+1} are identical, except that $r <_i s \wedge s <_{i+1} r$. By Lemma 6, O_{i+1} is also a legal serialization of O. It then follows from the hypothesis that r and s commute and from Lemma 8 that O_i and O_{i+1} are convergent.

If $O_{i+1} \neq O_j$, we continue the construction by finding an adjacent pair of operations whose order is different in O_{i+1}, O_j. By swapping the two operations, we obtain another legal serialization O_{i+2}. We can then continue to swap all such adjacent pairs until the last constructed serialization is equal to O_j. This is achievable since at every step the number of operation pairs in the corresponding newly constructed legal serialization whose orders are different in O_j decreases. At the end, the construction process results in a chain of legal serializations

where the first one is O_i and the last is O_j, and any consecutive pair of legal serializations is identical except for the order of an adjacent pair of elements. It then follows Lemma 8 that every consecutive pair of serializations in the chain is state convergent, thus $S_0(O_i) = S_0(O_j)$.

($\Rightarrow$:) (Proof by Contradiction.) We show that if a PoR consistent system $\mathscr{S}$ with a restriction set R is convergent, then for any pair of non-commuting shadow operations, there must exist a restriction confining the two operations in R. Since $\mathscr{S}$ is convergent, we know that for any R-order of $\mathscr{S}$, any pair of causal legal serializations of that R-order are convergent. We assume by contradiction that there exist two shadow operations u, v such that they don't commute and $r(u,v) \notin R$. By the definition of commutativity, there exists a state S such that $S+u+v \neq S+v+u$. We can find a state S_0 and a sequence of shadow operations of $\mathscr{S}$, $O'(P,<)$, such that $S_0(O') = S$. Then, we can construct a R-order $O(U,\prec)$, where

- $U = P \cup \{u, v\}$;
- for any pair of operations in P, m and n, $m < n \Longleftrightarrow m \prec n$;
- for any operation m in P, $m \prec u$ and $m \prec v$.

It follows from the above construction that u, v are the maximal elements of O. It follows from the definition of causal legal serialization (Definition 3) that we can construct two causal legal serializations L_1 and L_2 of O such that $L_1 = O' + u + v$ and $L_2 = O' + v + u$. As $S_0(L_1) = S_0(O' + u + v)$, $S_0(L_1) = S_0(O') + u + v$. It follows from $S_0(O') = S$ that $S_0(L_1) = S + u + v$. By a similar argument, $S_0(L_2) = S + v + u$. It then follows from $S+u+v \neq S+v+u$ that $S_0(L_1) \neq S_0(L_2)$. As L_1 and L_2 are not convergent, $\mathscr{S}$ is not convergent. Contradiction is found. □

3.2 Invariant preservation

In RedBlue consistency, the methodology for identifying restrictions imposed on RedBlue orders for maintaining invariants is to check if a shadow operation is invariant safe or not (meaning whether it can potentially violate invariants when executed against a different state from the one that it was generated from). If not, to avoid invariant violations, the generation and replication of all non-invariant safe shadow operations must be coordinated. However, we observed that for some non-invariant safe shadow operations u, the corresponding violation only happens when a particular subset of non-invariant safe shadow operations (including u) are not partially ordered. Therefore, to eliminate all invariant violating executions with a minimal amount of coordination, we need to precisely define, for each violation, the minimal set of non-invariant safe shadow operations that are involved. We call this set an `invariant-conflict operation set`, or short, `I-conflict set`. After these are identified, preserving invariants only requires adding a single restriction over any two shadow operations from each `I-conflict set` so that the concurrent violating executions will be eliminated from all admissible partial orders. We formally define `I-conflict sets` as follows.

Definition 9 (Invariant-conflict operation set)**.** *A set of shadow operations G is an* invariant-conflict operation set *(or short,* I-conflict set*), if the following conditions are met:*

- $\forall u \in G$, *u is non-invariant safe;*
- $|G| > 1$*;*
- $\forall u \in G$, $\forall$ *sequence P consisting of all shadow operations in G except u, i.e., $P = (G \setminus \{u\}, <)$,* $\exists$ *a reachable and valid state S, s.t. $S(P)$ is valid, and $S(P+u)$ is invalid.*

In the above definition, the last point asserts that G is minimal, i.e., removing one shadow operation from it will no longer lead to invariant violations. We will use the following example to illustrate the importance of minimality. Imagine that we have an auction on an item i being replicated across three sites such as US, UK and DE, and having initially a 5 dollar bid from *Charlie*. Suppose also that three shadow operations, namely, $placeBid'(i, Bob, 10)$, $placeBid'(i, Alice, 15)$, and $closeAuction'(i)$ are accepted concurrently at the three locations, respectively. After applying all of them against the same initial state at every site, we end up with an invalid state, where *Charlie* rather than *Bob* and *Alice* won the auction. This invariant violating execution involves three concurrent shadow operations, but one of the two bid placing shadow operations is not necessary to be included in G, as even after excluding the request from either *Bob* or *Alice*, the violation still remains. This is refleced in Definition 9, according to which $\{placeBid', closeAuction'\}$ is an `I-conflict set`, while $\{placeBid', placeBid', closeAuction'\}$ is not. Intuitively, avoiding invariant violations is to prevent all operations from the corresponding `I-conflict` set from running in a coordination-free manner. The minimality property enforced in the `I-conflict set` definition allows us to avoid adding unnecessary restrictions.

Based on the above definition, we formulate the invariant preservation property into the following theorem.

Theorem 10. *Given a PoR consistent system $\mathscr{S}$ with a set of restrictions $R_{\mathscr{S}}$, for any execution of $\mathscr{S}$ that starts from a valid state, no site is ever in an invalid state, if the following conditions are met:*

- *for any of its* `I-conflict set` *G, there exists a restriction $r(u,v)$ in $R_{\mathscr{S}}$, for at least one pair of shadow operations $u, v \in G$; and*
- *for any pair of shadow operations u and v, $r(u,v)$ in $R_{\mathscr{S}}$ if u and v don't commute.*

We prove the invariant preservation theorem by contradiction as follows:

Proof: We assume by contradiction that invariant violations are possible with a sufficient set of restrictions $R_{\mathscr{S}}$ in place. Let E be an invariant violating execution of $\mathscr{S}$ and $O(U,\prec)$ be a smallest R-order of E that triggers the violation. Let $O_i(U,<)$ be a causal legal serialization of R-order O at site i. As O_i violates the corresponding invariant, $S_0(O_i)$ is invalid. If U is empty, then $S_0(O_i) = S_0$, and O_i is in a valid state. This violates the assumption that O_i is in an invalid state. The theorem is proved.

Then we consider that U is non-empty. Let G be the set of shadow operations that are maximal according to $O(U,\prec)$, i.e., $G \subset U$, and $u \in G \Leftrightarrow \nexists v \in U$ s.t. $u \prec v$. The fact that U is not empty implies that G is not empty as well.

As follows, we will prove that G is an `I-conflict set`.

We first consider the case that G contains invariant-safe shadow operations. Let v be such an invariant safe shadow operation in G. If v is not the last operation in O_i, then it follows from Lemma 6 and 7 that we can swap v and any shadow operation u, s.t. $u \in G$, $u \neq v$ and $v < u$. This swapping process terminates when it produces a new legal serialization O_j of the R-order O, where v appears as the last operation, i.e., $O_j = O'_j + v$. It also follows from the assertion that any pair of shadow operations that are not ordered in $\prec$ commute w.r.t each other and Lemma 8 that $S_0(O_j) = S_0(O_i)$. As $S_0(O_i)$ is invalid and $S_0(O'_j) + v = S_0(O_j)$, $S_0(O'_j) + v$ is invalid. It then follows from the fact that v is invariant safe and the invariant safe shadow operation definition (Definition **??**) that the state before applying v must be invalid, i.e., $S_0(O'_j)$ is not valid. This implies that there exists a smaller R-order $O'(U \setminus \{v\}, \prec)$ than $O(U,\prec)$ that triggers the corresponding invariant. It contradicts with our assumption that O is a smallest R-order observing invalid state. Therefore, we only need to analyze the case when all shadow operations in G are non-invariant safe. The first condition of the `I-conflict set` definition is met.

We continue by checking if $|G| = 1$, i.e., G contains only a single non-invariant safe shadow operation. Let v be that operation. $O_i = O'_i + v$. As O_i is in an invalid state, $S_0(O'_i + v)$ is invalid. It follows from the assumption that O_i is the causal legal serialization of O at site i (where the generator of v was executed) and the correct shadow operation definition (Definition **??**) that the state $S_0(O'_i)$, which v was created from, is also invalid. By similar logic as above, there exists a smaller invariant violating R-order than O, and contradiction is found. As a result, $|G| > 1$. The second condition of the `I-conflict set` definition is met.

Finally, we check if G also meets the third condition of the `I-conflict set` definition. Let $O'_i(U \setminus G, <)$ be a prefix of O_i excluding all operations of G from O_i. Let $S = S_0(O'_i)$. $\forall u \in G$, we can construct a legal serialization O_j of O such that $O_j = O'_i + T + u$, where T is a sequence consisting of all shadow operations in $G \setminus \{u\}$. It also follows from Lemma 6, 7, Lemma 8 and the assertion that any pair of unordered shadow operations commute that $S_0(O_i) = S_0(O_j)$. Since the underlying R-order O is a smallest R-Order violating the corresponding invariant, S and $S(T)$ are valid, and $S(T+u)$ is invalid.

As G meets all three conditions presented in the `I-conflict set` definition (Definition 9), G is an `I-conflict` set. It then follows from the assertion in the invariant preserving theorem that for any `I-conflict` set, there exists a restriction defined over one pair of shadow operations in the set so that it is impossible to have all shadow operations in G to not be ordered w.r.t each other in the R-order O. Therefore, G cannot be a maximal element set of the R-order O. Contradiction is found. □

References

[1] Li, C., Leitão, J. a., Clement, A., Preguiça, N., Rodrigues, R., and Vafeiadis, V. Automating the Choice of Consistency Levels in Replicated Systems. In *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference* (Berkeley, CA, USA, 2014), USENIX ATC'14, USENIX Association, pp. 281–292.

[2] Li, C., Porto, D., Clement, A., Gehrke, J., Preguiça, N., and Rodrigues, R. Making Geo-replicated Systems Fast As Possible, Consistent when Necessary. In *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation* (Berkeley, CA, USA, 2012), OSDI'12, USENIX Association, pp. 265–278.

[3] Shapiro, M., Preguiça, N., Baquero, C., and Zawirski, M. A Comprehensive Study of Convergent and Commutative Replicated Data Types. Tech. Rep. 7506, INRIA, Jan. 2011.