Verifying Eventual Consistency of Optimistic Replication Systems

Ahmed Bouajjani Constantin Enea Jad Hamza
LIAFA, Univ Paris Diderot, Sorbonne Paris Cité, Paris, France.
{abou,cenea,jhamza}@liafa.univ-paris-diderot.fr

Abstract

We address the verification problem of eventual consistency of optimistic replication systems. Such systems are typically used to implement distributed data structures over large scale networks. We introduce a formal definition of eventual consistency that applies to a wide class of existing implementations, including the ones using speculative executions. Then, we reduce the problem of checking eventual consistency to reachability and model checking problems. This reduction enables the use of existing verification tools for message-passing programs in the context of verifying optimistic replication systems. Furthermore, we derive from these reductions decision procedures for checking eventual consistency of systems implemented as finite-state programs communicating through unbounded unordered channels.

Categories and Subject Descriptors Theory of Computation [*Logics and meanings of programs*]: General

Keywords message passing concurrency, model checking, static program analysis

1. Introduction

Optimistic data replication is a key technology for achieving high availability and performance in distributed systems. It consists of a set of techniques for maintaining multiple copies of the same data, called *replicas*, on different sites in a large-scale network. Many interactive collaborative applications, ranging from social networks to collaborative spaces and online shops, use this technology, in order to increase the quality of their services. In fact, they use this technology even though the latter lets replicas temporarily diverge, and thus users may see inconsistent data from time to time. This is due to the fact that ensuring strong consistency (i.e., making all replicas always consistent) – which can only be achieved by synchronizing all replicas after each update – is practically infeasible, and therefore users generally prefer high responsiveness and availability to strong consistency. In fact, systems implementing optimistic data replication, called *optimistic replication systems* (ORS), can only ensure weak consistency notions, and one of the important issues in this context is to determine what are precisely these notions. By the CAP theorem [10], ensuring sequential consistency or linearizability together with availability and partition tolerance is not possible, and thus, the correctness criteria adopted for ORS must in general be weaker than these ones (that are more suitable for shared memory systems). One of the most popular correctness criteria for ORS (and which is in some sense the weakest that can be accepted) is *eventual consistency*. Many ORS that are widely used in practical applications, e.g., Amazon Simple Storage Service and Google App Engine Datastore, (are supposed to) satisfy eventual consistency.

Optimistic replication systems, such as those mentioned above, are extremely complex and hard to get right, and therefore, there is a real need in developing automated formal methods for reasoning about their behaviors and verifying their correctness w.r.t. criteria such as eventual consistency. However, the first issue to be addressed to achieve that is defining precisely and formally these criteria that are still not well understood in general. For instance, many works adopt different variants of what is called eventual consistency. Then, the second issue to address is (1) investigating the limits of decidability and complexity bounds for checking these criteria, and (2) designing effective algorithmic methods for their verification. In this paper, we address both of these semantical and algorithmic issues for eventual consistency. We introduce a formal definition of eventual consistency that is applicable to a wide class of ORS, including systems with speculative executions and roll-backs, and we provide an approach for its automatic verification based on effective reductions to reachability and model-checking queries.

In the following, we explain in more details the context and the nature of our contributions.

Optimistic replication systems: Many variants of ORS have been defined, e.g., [2, 7, 15–18, 20, 23]. In general, they are based on the following scenario: (1) users submit operations (i.e., self-contained updates or queries) to one of the sites, (2) operations are immediately applied to the local replica to let users continue working based on the effect of those operations, and (3) in the background, sites exchange and apply remote operations. Actually, this is the scenario used by *operation transfer systems*. There also exist *state transfer systems*, where the sites exchange the contents of their replicas. Although they may use different techniques, state transfer systems can be emulated by operation transfer systems, where the allowed operations are only to read and overwrite an entire object [20]. Therefore, we focus in this paper on operation transfer systems.

The communication between sites is usually based on *epidemic propagation*, which allows every operation to be communicated to all sites independently of the communication topology. When applying remote operations several critical components come into play (see Saito and Shapiro [19] for a survey):

- *scheduling policies*, used to order operations in a way expected by users, and to make sites eventually agree on some ordering, (a generic scheduling policy is to label operations by timestamps and execute them in the order of their timestamps),

POPL '14, January 22–24, 2014, San Diego, CA, USA.

http://dx.doi.org/10.1145/2535838.2535877

- *conflict detection*, used to detect operations that are submitted concurrently by users to different sites and touch the same data (e.g., two requests to a room-booking system that concern the same room and the same time slot),
- *conflict resolution*, used to define the effect of a set of conflicting operations (e.g., in the room-booking system, the system might require that each request comes with some alternative time slots, and in the case of two conflicting requests, one will get an alternative time slot),
- *commitment protocols*, used to make sites agree about the order in which they execute the operations.

In particular, a large class of ORS, e.g., Bayou [23], C-Praxis [18], IceCube [16], Telex [2], use *speculative executions*, i.e., operations are tentatively applied as they are received from the user or from the other sites. In such systems, the sites may receive the operations in different orders and thus, they may have to repeatedly roll-back some operations as they gradually converge towards a final order (for example, in the case of Bayou, the final order is decided by a designated primary site).

Eventual consistency: In its simplest formulation[1], eventual consistency requires that if the users stop submitting operations, then all the sites will eventually reach a consistent state (i.e., they agree on the way operations should be executed). However, as mentioned in Burckhardt et al. [6], this formulation is too loose and the reason is twofold. First, this definition does not impose some notion of correctness for the operations executed by the system, i.e., the fact that they should satisfy some well-defined specification. Second, this property offers no guarantees when the system never stops, i.e., when the users continuously submit new operations. For that, eventual consistency should take into account infinite executions of the system involving infinitely many operations.

In fact, works investigating the formal definition of eventual consistency (and correctness criteria for distributed data structures in general) are still very rare. To our knowledge, Burckhardt et al. [6] is the first attempt to provide a formal framework for reasoning about eventual consistency. However, the proposed formalization in that paper does not allow to reason about ORS, that use speculative executions and roll-backs (as the authors of that paper mention). So, the first issue that we address in this paper is providing a general formal definition of eventual consistency, that is applicable to a wider class of ORS.

We define eventual consistency as a property over *traces*. A trace models the view of an external observer of the system; it is a set of sequences of operations, where one sequence consists of all the operations submitted to one site. It abstracts away implementation details such as the messages exchanged between sites.

Eventual consistency is defined as the composition of a safety property that specifies the correct effects of the operations, and a liveness property guaranteeing that sites will eventually agree on the order in which the operations should be executed. Let us look closer to each of these components.

Safety: Following the scenario described earlier, the return value of an operation o submitted to some site N depends on (1) the operations received and scheduled by N before o, and the order in which these operations are executed, (2) the conflict detection and conflict resolution policies applied by N, and (3) the behavior of the executed operations[2].

A trace is called *safe* iff the return values of all its operations are correct in a sense described hereafter. First, the correctness is defined with respect to a *specification* that, roughly, models the expected outcome of executing a poset of operations on a single site. Concretely, a poset of operations models a schedule, where incomparable elements are considered to be in conflict (i.e., submitted concurrently to different sites) and executing a poset of operations involves the actual implementations of the operations together with the conflict resolution policy (that defines the effect of concurrently submitted operations). A specification S associates return values of operations with posets of operations. Intuitively, the return value r of an operation o is associated with some poset ρ if o returns r whenever it is executed after the poset of operations ρ.

More precisely, a trace τ is safe w.r.t a specification S iff for each operation o, there exists a poset $li[o]$ of operations in τ, which is associated by S with the return value of o. The poset $li[o]$ is called the *local interpretation* of o. Additionally, because of physical constraints, we define an *executed-before* relation eb over the operations in the trace such that $(o', o) \in eb$ iff o' belongs to the local interpretation of o, and we require that the union of eb with the program order relation is an acyclic relation. Concretely, $(o', o) \in eb$ iff o is an operation submitted to some site N and o' is scheduled by N before o while the local interpretation $li[o]$ models the result of applying the scheduling and conflict detection policies.

Note that, for ORS that use speculative executions and roll-backs, the local interpretations associated with two operations are arbitrarily different, even if the two operations are submitted to the same site. (The issue of convergence is left to the liveness part.)

Liveness: The liveness part of eventual consistency requires that there exists some partial order relation gi (for global interpretation) over all the operations in an infinite trace, such that it is possible to choose some local interpretations satisfying the safety property, which *converge* towards gi. The convergence is formally stated as follows: for any prefix P of gi (a prefix of a poset is a restriction of the poset to a downward closed subset) there exists only finitely many local interpretations for which P is not a prefix. This corresponds to the informal definition given in Saito and Shapiro [19].

Verifying eventual consistency: After providing a formal and general definition of eventual consistency, we address the problem of checking whether a given ORS satisfies eventual consistency w.r.t some given specification. Our aim is twofold. First, we investigate the question of defining a generic algorithmic verification method for eventual consistency which is independent from the class of programs used for the implementation. This is actually a hard problem since, regardless of the complexity of the implementation, eventual consistency requires reasoning about nested quantifiers over (infinite) partial orders. Then, our second goal is to derive from the generic verification method decidability and complexity results for particular, as large as possible, classes of implementations. Let us then examine each of these points.

From eventual consistency to model-checking problems: We show that the static verification of (the safety and liveness parts of) eventual consistency can be reduced to reachability and model checking problems. This reduction allows to use any existing tools for exact/approximate verification of concurrent (message-passing) programs in the context of verifying eventual consistency. The reduction is done by defining a monitor which, when composed in parallel with the system, reaches some specific error state or violates some temporal logic formula (in LTL extended with Presburger predicates) if and only if there exists an execution of the system that violates eventual consistency. The monitor is a sequential program that observes, in a centralized way, the operations submitted to all the sites in the network. Therefore, its observations are interleavings of operations performed by the ORS. Reasoning about such centralized observations is possible since we are concerned with the static (offline) verification of ORS. The monitor is the parallel composition of two monitors, one for checking the safety part in eventual consistency and one for checking the liveness part. These monitors

[1] Also called, quiescent consistency [12]

[2] In this paper, we make the simplifying assumption that replicas are copies of the same object. In general, a replica may be composed of copies of multiple objects but most replication systems manage each object independently.

are abstractly defined as state transition systems that take a step each time an operation is executed in the system. Actually, they can be seen as a way of instrumenting every operation performed by the system in order to obtain a program on which the reachability and model-checking queries mentioned above are applied.

For these reductions, we assume that the operations are instances of a finite set of *methods* with a finite set of possible input/return values. However, system implementations can still be very complex: users are allowed to submit unboundedly many operations, and systems can use unbounded domains for timestamps or unbounded channels for communication.

The constructions we provide are based on subtle arguments. Indeed, one of the issues is that return values of operations depend on arbitrarily large sets of prior operations, and unbounded number of possible orders between these operations. The effect of each operation depends on the local view of the site N to which it is submitted. The local view consists of the set of operations known by N and the order in which N *believes* they were executed by other sites. In fact, when considering the safety part, we only need to check that, for each operation, such a local view exists and it is consistent with the return value of the operation. Unlike for the liveness part, we do not need to check the convergence of the local views toward a global view – which is a partial order over the infinite set of operations in the trace. In order to know if such local views exist, it is sufficient to count the number of *methods* previously issued. A careful analysis of the definition allows to show that it is possible to count only up to some finite bound, and thus only consider the minimal posets in the specification. Therefore, the problem of finding a violation for the safety property can be reduced to some bounded counting argument where the monitor counts the number of times the system executes each of the methods and then compares the counter values to the number of methods in the minimal posets of the specifications. This raises the problem of computing the number of methods in the minimal posets of the specification. In order to address this problem, we need to consider some concrete formalism for describing the specifications.

Therefore, we introduce a class of automata for describing specifications of ORS. Essentially, each poset of operations in the specification is abstracted into a sequence of multisets of methods (each method in these multisets corresponds to the method instantiated by an operation in the poset) which is then recognized by an automaton, called *multiset automaton*, where the transitions, instead of being labeled by symbols as in the case of automata over words, are labeled by Presburger formulas. The sequence of multisets corresponds to a decomposition of the poset in *levels*, used in algorithms for parallel tasks scheduling [22], where a level is a set of elements for which the length of the longest path to a maximal element is the same. These automata offer a good compromise between tractability and expressive power. In particular, they allow to define a wide-class of specifications, e.g., specifications of replication systems that use the Last Writer Wins conflict resolution policy [15] or commutative conflict resolution policies like in the case of the CRDTs [7, 20]. We then prove that for specifications recognized by multiset automata, it is possible to effectively compute the number of methods in the minimal posets.

For the liveness part of eventual consistency, we consider again a parallel composition between the system and a monitor that counts the number of times the system executes each of the methods. We show that the problem of finding a violation can be reduced to the problem of checking that the monitor reaches a state where its counters satisfy some specific property, and from there on, the methods executed by the system return only some particular set of values. The property that the counters should satisfy is quite complex; it characterizes the number of methods occurring finitely-often in the limit of an infinite sequence of posets belonging to the specification. However, we show that for specifications recognized by multiset automata, this property is definable by a Presburger formula, which can be effectively computed.

Decidability and complexity: Based on these reductions, we define decision procedures for checking eventual consistency of ORS, which are defined as the parallel composition of a fixed set of boolean programs communicating through unbounded unordered channels. Indeed, we consider that it is natural in our context to assume that channels are unordered since in general large-scale networks offer no guarantees on the order in which messages arrive even if they are sent by the same site. These decidability results are based on the fact that (1) this class of systems can be modeled by Vector Addition Systems with States (VASS, for short), since unbounded channels can be encoded as counters, and (2) the problems to which we reduce checking the safety, resp., the liveness, part of eventual consistency are decidable for VASS.

2. Preliminaries

Sets, relations: For a set A, $\mathcal{P}(A)$ denotes the set of all subsets of A. Let $R \subseteq A \times B$ be a relation. For any $y \in B$, $R^{-1}(y)$ denotes the set of elements $x \in A$ such that $(x, y) \in R$. The notation is extended to subsets of B as usual.

Sequences: Let Σ be a finite alphabet. The set of all finite, resp., infinite, sequences over Σ is denoted by Σ^*, resp., Σ^ω. Also, $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$. A set of sequences is called prefix-closed if the prefix of any sequence in the set is also in the set.

Posets: Let $(A, \leq)$ be a possibly infinite partially-ordered set (poset, for short). A path of length n in $(A, \leq)$ is a sequence of elements x_0,x_1,. . .,x_n such that for all $0 \leq i \leq n-1$, $x_i \leq x_{i+1}$.

The downward closure of $B \subseteq A$ w.r.t. $\leq$, denoted by $\downarrow^{\leq} B$, is the set of all elements in A smaller than B, i.e., $\downarrow^{\leq} B = \leq^{-1}(B)$. Similarly, the upward closure of $B \subseteq A$ w.r.t. $\leq$, denoted by $\uparrow^{\leq} B$, is the set of all elements in $x \in A$ such that $y \leq x$, for some $y \in B$. The superscript may be omitted when the partial order $\leq$ is clear from the context. We say that B is downward closed, resp., upward closed, w.r.t. $\leq$ iff $B = \downarrow B$, resp., $B = \uparrow B$.

The poset $(A_1, \leq_1)$ is called a *prefix* of $(A_2, \leq_2)$, denoted by $(A_1, \leq_1) \preceq (A_2, \leq_2)$, iff $A_1 \subseteq A_2$, $\leq_1$ is the intersection of $\leq_2$ and $A_1 \times A_1$, and A_1 is downward closed w.r.t. $\leq_2$.

We say that a poset $(A, \leq)$ is *prefix-founded* iff for all finite sets $B \subseteq A$, $\downarrow B$ is finite.

Labeled posets: A Σ-*labeled poset* is a triple $(A, \leq, \ell)$, where $(A, \leq)$ is a poset and $\ell : A \to \Sigma$ is a function that labels each element of A with a symbol in Σ. The set of all Σ-labeled posets is denoted by PoSet_Σ.

Given $\Sigma' \subseteq \Sigma$, the *projection* of a Σ-labeled poset $(A, \leq, \ell)$ over Σ' is the Σ'-labeled poset $(B, \leq', \ell')$, where B is the subset of A containing all the elements labeled by symbols in Σ', i.e., $B = \{a \mid \ell(a) \in \Sigma'\}$, and $\leq'$, resp., ℓ' are the projections of $\leq$, resp., ℓ over $B \times B$, resp., B.

The Parikh image of a possibly infinite Σ-labeled poset $\rho = (A, \leq, \ell)$ is the multiset $\Pi_\Sigma(\rho) : \Sigma \to \mathbb{N} \cup \{\omega\}$ mapping each symbol $a \in \Sigma$ to the number of elements x of A such that $\ell(x) = a$. If some symbol a occurs infinitely often then $\Pi_\Sigma(\sigma)(a) = \omega$. Note that the Parikh image of a labeled poset ρ can be also interpreted as the multiset of symbols in ρ. The notion is extended to sets of labeled posets as usual.

A labeled poset $\rho_1 = (A_1, \leq_1, \ell_1)$ is called a prefix of $\rho_2 = (A_2, \leq_2, \ell_2)$ iff the poset $(A_1, \leq_2)$ is a prefix of $(A_2, \leq_2)$ and $\ell_1(y) = \ell_2(y)$, for all $y \in A_1$. We also say that ρ_2 is a *completion* of ρ_1. This is denoted by $\rho_1 \preceq \rho_2$. Moreoever, if $\ell_2(A_2 \setminus A_1) \subseteq \Sigma'$ where $\Sigma' \subseteq \Sigma$, we say that ρ_2 is a Σ'-completion of ρ_1.

Two labeled posets $(A_1, \leq_1, \ell_1)$ and $(A_2, \leq_2, \ell_2)$ are isomorphic iff there exists a bijection $h : A_1 \to A_2$ such that for all

$x, y \in A_1$, $x \leq_1 y$ iff $h(x) \leq_2 h(y)$ and $\ell_1(x) = \ell_2(h(x))$. A set of labeled posets $\mathcal{A}$ is called *isomorphism-closed* iff any labeled poset isomorphic to some labeled poset in $\mathcal{A}$ belongs also to $\mathcal{A}$.

Functions: Let A and B be two sets. We denote by $A \to B$ the set of functions from A to B. For a function $f \in A \to B$, and $a \in A, b \in B$, $f[a \mapsto b]$ is the function which maps all elements $a' \neq a$ to $f(a')$, and which maps a to b. If $A = \{a_1, \ldots, a_n\}$, and $b_1, \ldots, b_n$ are (possibly equal) elements of B, we denote by $(a_1 \mapsto b_1, \ldots, a_n \mapsto b_n)$ the function $f \in A \to B$ which maps a_i to b_i for all $1 \leq i \leq n$. Likewise, for $b \in B$, we denote by $(a \in A \mapsto b)$ the function which maps every $a \in A$ to b.

For functions $f : A \to \mathbb{N}$, and $g : A' \to \mathbb{N}$ with $A' \subseteq A$, $f + g : A \to \mathbb{N}$ maps each $a' \in A'$ to $(f + g)(a') = f(a') + g(a')$ and each $a \in A \setminus A'$ to $(f + g)(a) = f(a)$.

3. Modeling optimistic replication systems

A concrete execution of an ORS is modeled as a trace, that abstracts away many implementation details. A trace is a poset of operations, where all the operations submitted to the same site are totally ordered. We suppose that operations are instances of a fixed set of *methods*. For static verification of eventual consistency, an ORS is viewed as a prefix-closed set of *sequential observations*. These are sequences of pairs of the form (N, o), where N is a site name and o is an operation. Such a pair denotes an operation o submitted to site N. The trace represented by a sequential observation η contains all the operations in η such that any two operations submitted to different sites are incomparable and the order between any two operations submitted to the same site is consistent with the order in which these two operations appear in the sequence η.

Let $\mathbb{M}$ be the set of method names and $\mathbb{D}$ the domain of return values. For simplicity, we suppose that input parameters are encoded in the method names. A *method synopsis* is a pair $m \triangleright r$, where $m \in \mathbb{M}$ and $r \in \mathbb{D}$. We denote by $\mathbb{M} \triangleright \mathbb{D}$ the set of all method synopses. An *operation* o is a pair (i, a) formed of an identity $i \in \mathbb{N}$ and a method synopsis $a \in \mathbb{M} \triangleright \mathbb{D}$. We denote by $\mathbb{O}$ the set of all operations. Given a method synopsis $a = m \triangleright r$, we let $\mathsf{meth}(a) = m$, $\mathsf{rval}(a) = r$, and given an operation $o = (i, m \triangleright r)$, we let $\mathsf{id}(o) = i$, $\mathsf{meth}(o) = m$, $\mathsf{rval}(o) = r$, and $\mathsf{syn}(o) = m \triangleright r$. An operation $o = (i, m \triangleright r)$ may be called an *m-operation* or an *$m \triangleright r$-operation*. Also, given a set of method names $M \subseteq \mathbb{M}$ (resp., a set of method synopses $Y \subseteq \mathbb{M} \triangleright \mathbb{D}$) o is called an *M-operation* (resp., *Y-operation*) iff $m \in M$ (resp., $m \triangleright r \in Y$).

A *trace* τ is a (possibly infinite) poset (O, po), where $O \subseteq \mathbb{O}$ and po, called the *program order*, is a disjoint union of a set of prefix-founded irreflexive total orders over O. We assume that a trace does not contain two operations with the same identity. The set of operations, resp., the program order, in a trace τ are denoted by O_τ, resp., po_τ.

Given a fixed set $\mathcal{N}$ of site names, a *sequential observation* (observation, for short) of an ORS is a possibly infinite sequence over $\mathbb{O}_\mathcal{N} = \mathcal{N} \times \mathbb{O}$ (we assume that such a sequence does not contain two operations with the same identity). A sequential observation models the view of a centralized sequential observer over the concrete executions of the replication system. Then, an *optimistic replication system* $\mathcal{I}$ is a state machine that produces a prefix-closed subset of $(\mathbb{O}_\mathcal{N})^\infty$. By an abuse of notation, the set of sequences produced by $\mathcal{I}$ is denoted also by $\mathcal{I}$.

Given an observation η, let O_η denote the set of operations that occur in η, i.e., the set of operations o for which there exists $N \in \mathcal{N}$ such that (N, o) occurs in η. Given an observation $\eta \in (\mathbb{O}_\mathcal{N})^\infty$, the *trace of* η is defined by $trace(\eta) = (O, po)$, where

- $O = O_\eta$ and
- $po = \{(o, o') \mid \exists N \in \mathcal{N}$ s.t. (N, o) occurs before (N, o') in $\eta\}$

Next, we give examples of ORS that we use throughout the paper.

Example 1 (One-value Register)**.** The One-Value Register (Register) maintains an integer register and supports the set of methods $\mathbb{M}_\mathsf{R} = \{\mathtt{wr}(i) \mid i \in \mathbb{N}\} \cup \{\mathtt{rd}\}$, where $\mathtt{wr}(i)$ assigns value i to the register and $\mathtt{rd}$ reads the current value of the register. A method $\mathtt{rd}$ can return any value from $\mathbb{N}$ while the methods $\mathtt{wr}(i)$ return some special value $\top$, i.e., the domain of return values is $\mathbb{D}_\mathsf{R} = \mathbb{N} \cup \{\top\}$. We thus have $\mathbb{M} \triangleright \mathbb{D}_\mathsf{R} = \{\mathtt{wr}(i) \triangleright \top, \mathtt{rd} \triangleright i \mid i \in \mathbb{N}\}$.

Example 2 (Multi-Value Register)**.** The Multi-Value Register [7, 14] (MV-Register) maintains an integer register and supports the same set of methods as the Register. A method $\mathtt{rd}$ can return any *set* of values from $\mathbb{N}$. Thus, the domain of return values of the MV-Register is $\mathbb{D}_{\mathsf{MV\text{-}R}} = \mathcal{P}(\mathbb{N}) \cup \{\top\}$ and its set of method synopses is $\mathbb{M} \triangleright \mathbb{D}_{\mathsf{MV\text{-}R}} = \{\mathtt{wr}(i) \triangleright \top, \mathtt{rd} \triangleright I \mid i \in \mathbb{N}$ and $I \subseteq \mathbb{N}\}$.

Example 3 (Observed-Remove Set)**.** The Observed-Remove Set [20] (OR-Set) maintains a set of integers over which one can apply the set of methods $\mathbb{M}_{\mathsf{OR\text{-}S}} = \{\mathtt{add}(i), \mathtt{rem}(i), \mathtt{lookup}(i) \mid i \in \mathbb{N}\}$, where $\mathtt{add}(i)$ adds the integer i to the set, $\mathtt{rem}(i)$ removes i from the set, and $\mathtt{lookup}(i)$ tests if the integer i is in the set. We assume that the methods $\mathtt{add}$ and $\mathtt{rem}$ return some fixed value $\top$, while $\mathtt{lookup}(i)$ can return 1 or 0. Thus, the set of return values is $\mathbb{D}_{\mathsf{OR\text{-}S}} = \{1, 0, \top\}$ and $\mathbb{M} \triangleright \mathbb{D}_{\mathsf{OR\text{-}S}} = \{\mathtt{add}(i) \triangleright \top, \mathtt{rem}(i) \triangleright \top, \mathtt{lookup}(i) \triangleright 1, \mathtt{lookup}(i) \triangleright 0 \mid i \in \mathbb{N}\}$.

4. Eventual consistency

In this section, we introduce a formal definition for eventual consistency, whose main artifacts are presented hereafter.

Specification: In the context of shared-memory, resp., ORS, the correctness of the operations associated to some object usually involves some mechanism of conflict resolution in order to define the effect of a set of operations executed by different threads, resp., submitted concurrently at different sites. In the case of shared-memory systems, the conflict resolution mechanism is usually specified by the notion of *linearizability* [13], which requires that the effect of a set of concurrent operations (that overlap in time) is the same as the one of a sequential execution of the same set of operations. In the context of ORS, where each site maintains its own copy of the object, some more general mechanisms of conflict resolution are required.

To specify both the sequential semantics of the operations and the conflict resolution mechanisms we use posets of operations instead of sequences of operations, as in the case of linearizable objects. We don't use a total order because, in general, it is unfeasible that all sites agree on a total order of operations, and sometimes even unnecessary, in case of commutative operations for instance. Moreover, the mechanisms used to detect conflicting (causally unrelated) operations are not always precise.

We assume that the specification can't distinguish between two posets that are identical (i.e., isomorphic) when ignoring the identities and the return values of the operations. The insensitivity to return values is motivated by the fact that, in real implementations, the sites exchange operations without their return values (in such systems, it is not expected that an operation returns the same value when executed at different sites).

A specification associates to each method synopsis $m \triangleright r$ an isomorphism-closed set of $\mathbb{M}$-labeled posets. The closure under isomorphism and the fact that we use labeled posets model the fact that the specification doesn't observe identities and return values. The fact that a poset ρ is associated to $m \triangleright r$ means that any site that sees the set of operations in ρ in that order reaches a state where the call to m produces the value r.

Definition 1 (Specification)**.** A *specification* is a function $S : \mathbb{M} \triangleright \mathbb{D} \to \mathcal{P}(\mathsf{PoSet}_\mathbb{M})$, where for each method synopsis $a \in \mathbb{M} \triangleright \mathbb{D}$, $S(a)$ is an isomorphism-closed set of $\mathbb{M}$-labeled posets.

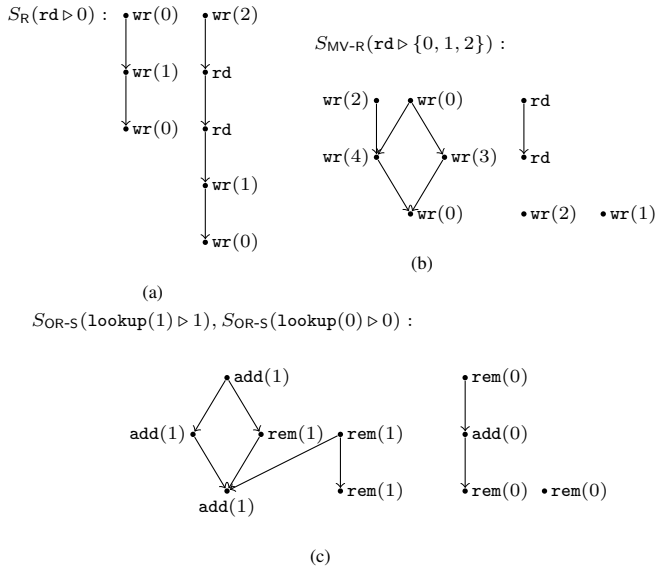


Figure 1: Examples of labeled posets belonging to (a) the specification of the Register (b) the specification of the MV-Register, and (c) the specification of the OR-Set. The order relations are defined by arrows: an arrow from an element x to some element y means that x is ordered before y. We omit arrows implied by transitivity.

We give several examples of specifications for the replication systems mentioned in the previous section.

Example 4 (Register specification)**.** The specification S_{R} of the Register is given by: (1) for every $a = \mathtt{wr}(i) \triangleright \top$, $S_{\mathsf{R}}(a)$ is the set of all $\mathbb{M}_{\mathsf{R}}$-labeled totally-ordered sets, and (2) for every $a = \mathtt{rd} \triangleright i$, $S_{\mathsf{R}}(a)$ is the set of all $\mathbb{M}_{\mathsf{R}}$-labeled totally-ordered sets where the maximal element labeled by a write is labeled by $\mathtt{wr}(i)$. Figure 1a contains two examples of totally-ordered sets in $S_{\mathsf{R}}(\mathtt{rd}\triangleright 0)$.

Example 5. [MV-Register specification] The specification $S_{\mathsf{MV\text{-}R}}$ of the MV-Register is defined by: (1) for any $a = \mathtt{wr}(i) \triangleright \top$, $S_{\mathsf{MV\text{-}R}}(a)$ is the set of all $\mathbb{M}_{\mathsf{R}}$-labeled posets and (2) for any $a = \mathtt{rd} \triangleright I$, $S_{\mathsf{MV\text{-}R}}(a)$ is the set of all $\mathbb{M}_{\mathsf{R}}$-labeled posets ρ such that $i \in I$ iff there exists a maximal element labeled by $\mathtt{wr}(i)$ in the projection of ρ over the elements labeled by write methods $\{\mathtt{wr}(i) \mid i \in \mathbb{N}\}$. Figure 1b contains an $\mathbb{M}_{\mathsf{R}}$-labeled poset in $S_{\mathsf{R}}(\mathtt{rd} \triangleright \{0,1,2\})$.

Example 6. [OR-Set specification] The specification $S_{\mathsf{OR\text{-}S}}$ is given by: (1) for any $a = \mathtt{add}(i) \triangleright \top$ or $a = \mathtt{rem}(i) \triangleright \top$, $S_{\mathsf{OR\text{-}S}}(a)$ is the set of all $\mathbb{M}_{\mathsf{OR\text{-}S}}$-labeled posets, and (2) for any $a = \mathtt{lookup}(i) \triangleright 1$, resp., $a = \mathtt{lookup}(i) \triangleright 0$, $S_{\mathsf{OR\text{-}S}}(a)$ is the set of all $\mathbb{M}_{\mathsf{OR\text{-}S}}$-labeled posets ρ such that the projection of ρ over the elements labeled by $\mathtt{add}(i)$ or $\mathtt{rem}(i)$ contains a maximal element labeled by $\mathtt{add}(i)$, resp., contains no maximal element labeled by $\mathtt{add}(i)$. Figure 1c contains an example of a labeled poset that belongs to both $S_{\mathsf{OR\text{-}S}}(\mathtt{lookup}(1) \triangleright 1)$ and $S_{\mathsf{OR\text{-}S}}(\mathtt{lookup}(0) \triangleright 0)$.

Local interpretation: The return value of some operation o submitted to some site N depends on the set of operations applied at N before o and on the the effect of applying the scheduling and conflict detection policies over this set of operations. Taken together they can be represented by a poset of operations, called the *local interpretation* of o and denoted by $li[o]$. Because of speculative executions, one has to consider a local interpretation for each operation o in the trace (the order in which known operations are executed can change at any time).

The local interpretations define another relation over the operations in the trace, called *executed-before* and denoted by eb. We say that some operation o' is executed before another operation o, i.e., $(o', o) \in eb$, iff o' belongs to the local interpretation of o. For example, Figure 2a pictures a trace of the Register, where the arrows define a possible eb relation. Note that the $\mathtt{wr}(1)$-operation is executed before the first occurrence of $\mathtt{rd} \triangleright 0$ but not before the second occurrence of $\mathtt{rd} \triangleright 0$.

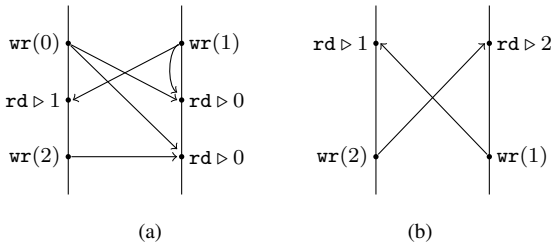


Figure 2: Traces of the Register. For simplicity, the return values of the $\mathtt{wr}$ operations and the ids of each operation are omitted. The program order is defined by the vertical lines, from top to bottom.

We say that the return value of some operation o is *correct* iff the labeled poset defined by $li[o]$, where every operation o' is labeled by $\mathsf{meth}(o')$, belongs to $S(\mathsf{syn}(o))$. Then, a trace τ is *safe* iff the return values of all operations in τ are correct.

For example, in the case of the first $\mathtt{rd} \triangleright 0$-operation in Figure 2a, one can choose to order the $\mathtt{wr}(1)$-operation before the $\mathtt{wr}(0)$-operation. This local interpretation defines an $\mathbb{M}_{\mathsf{R}}$-labeled poset, which belongs to the specification of the Register, $S_{\mathsf{R}}(\mathtt{rd} \triangleright 0)$. Similarly, one can show that all return values in Figure 2a are correct.

Because of physical constraints, eb must not create cycles together with the program order. For example, the trace in Figure 2b could be one of the Register if the relation eb is defined by the arrows in the figure (we assume that the initial value of the register is 0). However, this means that the site executing the operations in the left received a message from the other site containing the $\mathtt{wr}(1)$ operation and this message was created after $\mathtt{rd} \triangleright 2$ has finished. Thus, in real time, $\mathtt{rd} \triangleright 2$ has happened before the $\mathtt{wr}(2)$-operation, which contradicts the eb relation.

Global interpretation: The fact that the sites will eventually agree on the way operations should be executed is defined as a liveness property over infinite traces of the system. Given an infinite trace τ, we consider a partial-order over all the operations in τ, called the *global interpretation* and denoted by gi. The liveness property requires that the local interpretations defined for the operations in τ *converge* towards gi. More precisely, for any finite prefix P of gi, the number of local interpretations for which P *is not* a prefix is finite. Note that this implies that any operation o in the trace is executed before all operations in τ, except some finite set. A system that satisfies only this property will be called *weak eventually consistent*.

We will require that gi satisfies some sanity condition, i.e., that it is prefix-founded, which basically, means that every operation can be executed after only finitely many other operations.

For example, let us consider the infinite trace in Figure 3. We consider a global interpretation gi, which in this case is a total order, that arranges the write operations in a sequence of the form $(\,\mathtt{wr}(0)\,\mathtt{wr}(1)\,)^{\omega}$, and keeps the same order as in the trace for the $\mathtt{wr}(0)$ operations (resp., the $\mathtt{wr}(1)$ operations). In the following, we ignore the order between the $\mathtt{rd}$-operations because they are not important for justifying the correctness of the return values. We show that it is possible to choose local interpretations for the $\mathtt{rd} \triangleright 0$ operations such that the return values are correct, and such that the orders converge towards gi. In a similar way, this can be shown for all the other operations in the trace.

For each $\mathtt{rd} \triangleright 0$-operation o, the local interpretation $li[o]$ is the poset that consists of all the write operations that occur before o (i.e., if o is the ith occurrence of $\mathtt{rd} \triangleright 0$ then the poset $li[o]$ contains

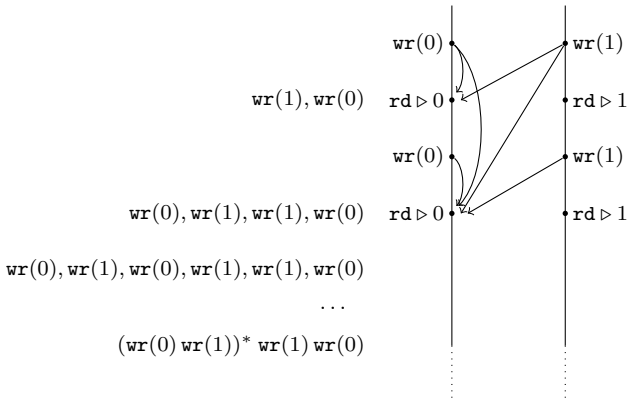


Figure 3: An infinite trace of Register, where one site executes $(\,\mathtt{wr}(0)\,\mathtt{rd}\triangleright 0\,)^\omega$ and another site executes $(\,\mathtt{wr}(1)\,\mathtt{rd}\triangleright 1\,)^\omega$.

the jth occurrence of $\mathtt{wr}(1)$ and $\mathtt{wr}(0)$, for all $j < i$), totally ordered in a sequence of the form $(\mathtt{wr}(0)\,\mathtt{wr}(1))^*\ \mathtt{wr}(1)\,\mathtt{wr}(0)$ consistent with the program order. The relation eb is pictured by arrows in Figure 3. Defined as such, the local interpretations converge towards the global interpretation gi.

We now give the definition of eventual consistency. For any poset $li[o] = (A, \leq)$ as above, $li_{\mathbb{M}}[o]$ denotes the labeled poset $li[o]$ where every operation is labeled by the corresponding method name in $\mathbb{M}$, i.e., $li_{\mathbb{M}}[o] = (A, \leq, \mathsf{meth})$.

Definition 2. [Safety, (Weak) Eventual consistency] A trace $\tau = (O, po)$ is called *eventually consistent* w.r.t. a specification S iff:

$$\exists\ gi \text{ an irreflexive partial order over } O$$
$$\forall\ o \in O\ \exists\ li[o] \text{ an irreflexive poset.}$$
$$\textsc{Gipf} \wedge \textsc{ThinAir} \wedge \textsc{Rval} \wedge \textsc{Eventual}$$

It is said to be *weak eventually consistent* w.r.t S iff EVENTUAL is replaced by WEAKEVENTUAL in the condition above (thus, gi and GIPF can be removed). Finally, it is said to be *safe* w.r.t S iff only the axioms THINAIR and RVAL are satisfied, i.e.,

$$\forall\ o \in O\ \exists\ li[o] \text{ an irreflexive poset.}$$
$$\textsc{ThinAir} \wedge \textsc{Rval}$$

The relation eb is defined by: $(o', o) \in eb$ iff $o' \in li[o]$

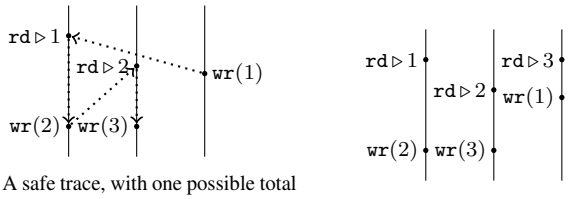
THINAIR	$eb \cup po$ is acyclic
RVAL	for all $o = (i, a) \in O$, $li_{\mathbb{M}}[o] \in S(a)$
GIPF	gi is prefix-founded
WEAKEVENTUAL	for all $o \in O$, $\{o' \mid (o, o') \notin eb\}$ is finite
EVENTUAL	for any finite prefix P of the poset (O, gi), $\{o \mid P \not\preceq li[o]\}$ is finite

Table 1: The list of axioms used in Definition 2.

Axioms THINAIR and RVAL are thus safety conditions, and ensure that the operations respect the specification S. Axiom WEAKEVENTUAL is a liveness condition, which ensures that eventually, every operation will be executed before all the other operations in the system. Axiom EVENTUAL is a stronger liveness condition which ensures that all nodes eventually agree on a (possibly partial) order in which to execute all the operations.

An ORS $\mathcal{I}$ is said to be (weak) eventually consistent, resp., safe, iff for every observation η of $\mathcal{I}$, $trace(\eta)$ is (weak) eventually consistent, resp., safe.

In the next sections, we consider the problem of verifying eventual consistency and **we assume that $\mathbb{M} \triangleright \mathbb{D}$ is finite**.



(a) A safe trace, with one possible total order e depicted with arrows

(b) An unsafe trace

Figure 4: Examples of traces of the Register

5. Safety

We consider the problem of checking that an ORS is safe and we prove that it can be reduced to a reachability problem. First, we show that in any total ordering over the operations of an *unsafe* trace τ, consistent with the program order, one can find an operation o such that there are not sufficiently many operations before o (in this total ordering) to construct a labeled poset belonging to its specification (i.e., $S(\mathsf{syn}(o))$). For example, for an unsafe trace of the Register, e.g., the trace in Figure 4b, with read and write operations, this means that, no matter in which order – consistent with the program order – it is read, there will always exist a read that returns a value not written by a previous write.

This allows us to define a monitor for checking the safety of a replication system $\mathcal{I}$, that records all the operations executed by $\mathcal{I}$, and stops with a negative answer at any time that it detects an operation o for which, with the recorded set of operations, it cann't build a labeled poset belonging to the specification of o. Actually, we prove that it is sufficient that the monitor only counts the number of times the system executes each of the methods in $\mathbb{M}$ (until some bound) and then, compare the counter values with the minimal vectors in the Parikh image of the specification.

The next lemma states the characterization of (un)safe traces.

Lemma 1. *A trace τ is safe w.r.t S iff there exists a prefix-founded total order e – called the* issue order *– on O_τ, consistent with po, such that for every operation o of τ, there exists a poset $(V_o, \leq_o)$, where V_o is a subset of $e^{-1}(o)$ and $(V_o, \leq_o, \mathsf{meth}) \in S(\mathsf{syn}(o))$.*

Proof. ($\Leftarrow$) For each o, define $li[o] = (V_o, \leq_o)$. This ensures that axiom RVAL holds. Moreover, the relations eb and po are both consistent with e, which implies that axiom THINAIR holds.

($\Rightarrow$) Conversely, assume that for each operation o, there exists $li[o]$ such that THINAIR and RVAL hold. Let e be any total order compatible with eb and po, which exists since $eb \cup po$ has no cycle. Figure 4a illustrates one such total order on a safe trace. Then, for each operation o, define $(V_o, \leq_o)$ as the poset $li[o]$. □

Lemma 1 implies that if τ is the trace of an observation η and it is unsafe, then there exists a prefix of η which ends in an operation whose return value is not correct, i.e., it is not possible to define a labeled poset that contains only operations in this prefix that belongs to the specification of o. This is stated in the following lemma.

Lemma 2. *Given an ORS $\mathcal{I}$, the following are equivalent:*

1. *there exists an observation $\eta \in \mathcal{I}$ such that $trace(\eta)$ is not safe*
2. *there exists an observation $\eta = \eta'(N, (\mathsf{id}, a)) \in \mathcal{I}$ such that there exists no poset $(V_o, \leq_o)$ whose elements are a subset of $O_{\eta'}$, such that $(V_o, \leq_o, \mathsf{meth}) \in S(a)$.*

Proof. Assume that (2) holds for $\eta = \eta'(N, (\mathsf{id}, a))$. Let $o = (\mathsf{id}, a)$ be the last operation of η. If η was safe, we would have a local interpretation $li[o]$ such that $li_{\mathbb{M}}[o] \in S(a)$. Note that the operations in $li[o]$ belong to $O_{\eta'}$, which contradicts (2). Thus, η is not safe and (2) implies (1).

Conversely, if we have an observation η such that $trace(\eta)$ is not safe, then by Lemma 1, there exists a prefix η_p of η such that η_p satisfies (2). Indeed, if there were no such prefix, the total order

on O_η induced by η would satisfy the condition of Lemma 1, which would contradict the fact that $trace(\eta)$ is not safe. □

The following corollary is a reformulation of Lemma 2 in terms of Parikh images. For any finite observation η, the $\mathbb{M}$-Parikh image of η is a function $\Pi_{\mathbb{M}}(\eta) : \mathbb{M} \to \mathbb{N}$, that maps each $m \in \mathbb{M}$ to the number of operations o in η with $m = \mathsf{meth}(o)$. The $\mathbb{M}$-Parikh image of a finite trace τ, denoted by $\Pi_{\mathbb{M}}(\tau)$, is defined similarly.

Corollary 1. *An optimistic replication system $\mathcal{I}$ is not safe iff there exists $\eta'(N, (\mathsf{id}, a)) \in \mathcal{I}$ such that $\Pi_{\mathbb{M}}(\eta') \notin \uparrow \Pi_{\mathbb{M}}(S(a))$.*

Given an observation η and an integer i, $\Pi^i_{\mathbb{M}}(\eta)$ is the Parikh image of η, where all the components larger than i are set to i, that is $\Pi^i_{\mathbb{M}}(\eta) = (m \in \mathbb{M} \mapsto min(i, \Pi_{\mathbb{M}}(\eta)(m)))$.

For each $a \in \mathbb{M} \triangleright \mathbb{D}$, let V_a be the set of minimal elements of $\Pi_{\mathbb{M}}(S(a))$ (w.r.t. the ordering relation over vectors of natural numbers), so that $\uparrow \Pi_{\mathbb{M}}(S(a)) = \uparrow V_a$, and let i_a be the maximum value appearing in the vectors of V_a. Let $i = \max\{i_a \mid a \in \mathbb{M} \triangleright \mathbb{D}\}$.

We remark that $\Pi_{\mathbb{M}}(\eta') \notin \uparrow \Pi_{\mathbb{M}}(S(a))$ is equivalent to the fact that $\Pi_{\mathbb{M}}(\eta')$ is *not* greater than one of the minimal elements $v_a \in V_a$. Moreover, since all the components of the vectors of V_a are smaller than i, $\Pi_{\mathbb{M}}(\eta') \notin \uparrow \Pi_{\mathbb{M}}(S(a))$ is equivalent to $\bigwedge_{v_a \in V_a} \Pi^i_{\mathbb{M}}(\eta') \not\geq v_a$.

In general, the sets V_a cannot be computed, but in Section 7, we give a class of specifications for which they can. We define a monitor $\mathcal{M}_{\mathtt{safe}}$, which counts all the methods it sees up to the bound i, and every time it reads a symbol $(N, (\mathsf{id}, a))$, it goes to an error state $q_{\mathtt{err}}$ iff the vector of methods seen is not larger than some $v_a \in V_a$.

Formally, $\mathcal{M}_{\mathtt{safe}}$ is a deterministic finite-state transition system (Q, Q_0, δ), where

- $Q = (\mathbb{M} \to \{0, \ldots, i\}) \cup \{q_{\mathtt{err}}\}$ is the finite set of states
- Q_0, the set of initial states, only contains $q_0 = (m \in \mathbb{M} \mapsto 0)$
- $\delta \subseteq Q \times \mathbb{O}_{\mathcal{N}} \times Q$ with
  - $(q, (N, (\mathsf{id}, a)), q_{\mathtt{err}}) \in \delta$ iff $\bigwedge_{v_a \in V_a} q \not\geq v_a$
  - $(q_1, (N, (\mathsf{id}, a)), q_2) \in \delta$ iff $\bigvee_{v_a \in V_a} q \geq v_a$ and $q_2 = q_1[\mathsf{meth}(a) \mapsto \min(q_1(\mathsf{meth}(a)) + 1, i)]$

Theorem 1 (Safety Monitoring)**.** *An optimistic replication system $\mathcal{I}$ is not safe if and only if the parallel composition $\mathcal{I} \| \mathcal{M}_{\mathtt{safe}}$ can reach the error state $q_{\mathtt{err}}$.*

6. Liveness

In this section, we give properties which characterize (weak) eventually consistent traces that will be used to define reductions of deciding (weak) eventual consistency to LTL model checking.

6.1 Weak Eventual Consistency

We first consider the case of weak eventual consistency because it is simpler while already showing some of the difficulties we have to solve. Moreover, for some systems, weak eventual consistency implies eventual consistency, for instance, when all the operations are commutative.

For an infinite trace τ to be weak eventually consistent there must exist some local interpretations which show that τ is safe but also, which ensure that each operation $o \in O_\tau$ is executed-before all the other operations, except for some finite set. The latter implies that any finite set of operations is executed-before every operation after some finite prefix. Thus, for any $a \in \mathbb{M} \triangleright \mathbb{D}$, if there are infinitely many a-operations in τ, then $S(a)$, the specification of a, must contain arbitrarily large posets. This property of $S(a)$ can be stated as a property of the Parikh image of $S(a)$ and this allows us to define a reduction of checking if some replication system $\mathcal{I}$ is weak eventually consistent to checking if $\mathcal{I}$ is safe and if the parallel composition of $\mathcal{I}$ with a monitor that counts the methods executed by $\mathcal{I}$ satisfies some LTL formula. Mainly, the temporal operators in this formula are used to identify the infinitely occurring method synopses in some execution.

In the following lemma, we characterize weak eventually consistent traces. To identify the infinitely occurring method synopses in some trace τ we use the following notation. Given $B \subseteq \mathbb{M} \triangleright \mathbb{D}$ and a finite trace τ_p, let $\tau_p B^\omega$ be the set of all traces τ which extend τ_p (i.e., τ_p is a prefix of τ) by an infinite set of B-operations such that there are infinitely many a-operations in τ, for each $a \in B$.

Lemma 3. *[Characterization of Weak Eventual Consistency] Given $B \subseteq \mathbb{M} \triangleright \mathbb{D}$ such that $\mathsf{meth}(B) = \{m_1, \ldots, m_k\}$, a trace $\tau \in \tau_p B^\omega$ is weak eventually consistent w.r.t. S if and only if*

- *τ is safe and*
- *$\forall a \in B. \forall n \in \mathbb{N}. \exists n_1, \ldots, n_k \geq n.$ $\Pi_{\mathbb{M}}(\tau_p) + (m_1 \to n_1, \ldots, m_k \to n_k) \in \Pi_{\mathbb{M}}(S(a))$*

Proof. ($\Rightarrow$) If τ is eventually consistent, then for each operation $o \in O_\tau$, there exists a local interpretation $li[o]$ such that the axioms ThinAir, Rval, and WeakEventual hold.

Let $a \in B$, and $n \in \mathbb{N}$. Since there are infinitely many a-operations in τ, we deduce from axiom WeakEventual that there exists one, noted o, such that $li[o]$ (or equivalently, $eb^{-1}(o)$) contains the operations of τ_p and at least n additional m-operations for each $m \in \mathsf{meth}(B)$. From axiom Rval, we know that $li_{\mathbb{M}}[o] \in S(a)$, which shows that there exists $n_1, \ldots, n_k \geq n$ such that $\Pi_{\mathbb{M}}(\tau_p) + (m_1 \to n_1, \ldots, m_k \to n_k) \in \Pi_{\mathbb{M}}(S(a))$.

($\Leftarrow$) Since τ is safe, there exists an issue order e over O_τ that satisfies the conditions in Lemma 1. In the following, we consider that the operations in τ are totally ordered according to e.

For each $n \in \mathbb{N}^*$, let o_n be the last operation in τ, which occurs after at most n m_i-operations for each $m_i \in \mathsf{meth}(B)$. For each $a \in B$ and $n \in \mathbb{N}^*$, let $n^a_1, \ldots, n^a_k \geq n$ such that

$$\Pi_{\mathbb{M}}(\tau_p) + (m_1 \to n^a_1, \ldots, m_k \to n^a_k) \in \Pi_{\mathbb{M}}(S(a))$$

and let o^a_n be the first operation in τ such that the prefix τ' of τ that ends in o^a_n, satisfies

$$\Pi_{\mathbb{M}}(\tau') \geq \Pi_{\mathbb{M}}(\tau_p) + (m_1 \to n^a_1, \ldots, m_k \to n^a_k). \quad (1)$$

The existence of o^a_n is ensured by the fact that τ contains infinitely many a-operations, for each $a \in B$.

For all $n \in \mathbb{N}^*$ and a-operation o in τ between o^a_n and o^a_{n+1}, the local interpretation $li[o] = (V_o, \leq_o)$ is defined as follows. The set V_o consists of all the operations of τ_p, all the m_i-operations before o_n, and some m_i-operations before o s.t.

$$\Pi_{\mathbb{M}}(V_o) = \Pi_{\mathbb{M}}(\tau_p) + (m_1 \to n^a_1, \ldots, m_k \to n^a_k).$$

This is possible because, for each $m_i \in \mathsf{meth}(B)$, there are at most n m_i-operations before o_n, and o occurs after o^a_n, that satisfies (1).

Now, since $\Pi_{\mathbb{M}}(V_o) \in \Pi_{\mathbb{M}}(S(a))$, there exists a partial order $\leq_o$ over the set V_o such that $(V_o, \leq_o, \mathsf{meth}) \in S(a)$. For the finite number of a-operations o that occur in τ before o^a_1, we use the local interpretations whose existence is ensured by the safety of τ.

Since both eb and po are consistent with the total order e, axiom ThinAir holds. For each operation, we have chosen $li[o]$ so that axiom Rval holds. Moreover, for all $n \in \mathbb{N}^*$, each operation o that occurs before o_n is executed-before all operations, except for a finite set – those that precede some o^a_n with $a \in B$. Thus, eb satisfies axiom WeakEventual, which concludes the proof. □

By Lemma 3, an ORS $\mathcal{I}$ violates weak eventual consistency iff it violates safety or if it produces a trace in $\tau_p B^\omega$, for some τ_p and $B \subseteq \mathbb{M} \triangleright \mathbb{D}$ with $\{m_1, \ldots, m_k\} = \mathsf{meth}(B)$, such that there exists a method synopsis $a \in B$ satisfying:

$$\exists n \in \mathbb{N}. \forall n_1, \ldots, n_k \geq n. \quad \Pi_{\mathbb{M}}(\tau_p) + (m_1 \to n_1, \ldots, m_k \to n_k) \notin \Pi_{\mathbb{M}}(S(a)) \quad (2)$$

Given $B \subseteq \mathbb{M} \triangleright \mathbb{D}$ and $a \in B$, let

$$\Pi_{\mathsf{notWEC}}(S,B,a) = \{v \mid \exists n \in \mathbb{N}. \forall n_1,\ldots,n_k \geq n. \\ v + (m_1 \to n_1,\ldots,m_k \to n_k) \notin \Pi_{\mathbb{M}}(S(a))\}.$$

Then, (2) can be rewritten as $\Pi_{\mathbb{M}}(\tau_p) \in \Pi_{\mathsf{notWEC}}(S,B,a)$.

Like for safety, we construct a monitor $\mathcal{M}_{\mathtt{live}}$, which counts the methods executed by the ORS, but this time without any bound. Finding a violation of weak eventual consistency reduces to finding a $B \subseteq \mathbb{M} \triangleright \mathbb{D}$ and a finite execution η_p in the monitored system $\mathcal{I} \| \mathcal{M}_{\mathtt{live}}$, such that $\bigvee_{a \in B} \Pi_{\mathbb{M}}(\eta_p) \in \Pi_{\mathsf{notWEC}}(S,B,a)$ and η_p can be extended by only using B-operations as well as infinitely many a-operations, for each $a \in B$. The latter can be checked using LTL model checking by adding to each state of the monitor, a register recording the method synopsis of the last transition.

Formally, $\mathcal{M}_{\mathtt{live}}$ is a transition system (Q, Q_0, δ), where

- $Q = (\mathbb{M} \to \mathbb{N}) \times \mathbb{M} \triangleright \mathbb{D}$,
- I is the set of initial states: $(q_0, a) \in I$ iff $q_0 = (m \in \mathbb{M} \mapsto 0)$ and $a \in \mathbb{M} \triangleright \mathbb{D}$,
- $\delta \subseteq Q \times \mathbb{O}_{\mathcal{N}} \times Q$ where $((q_1, b), (N, (\mathsf{id}, a)), (q_2, a)) \in \delta$ iff $q_2 = q_1[\mathsf{meth}(a) \mapsto q_1(\mathsf{meth}(a)) + 1]$ and $a, b \in \mathbb{M} \triangleright \mathbb{D}$.

Now, given a replication system $\mathcal{I}$, we consider the monitored system $\mathcal{I} \| \mathcal{M}_{\mathtt{live}}$ defined as the parallel composition of $\mathcal{I}$ and $\mathcal{M}_{\mathtt{live}}$. Define the following LTL formula:

$$\varphi_{\mathsf{notWEC}} = \bigvee_{B \subseteq \mathbb{M} \triangleright \mathbb{D}} \bigvee_{a \in B} \Diamond(\Pi_{\mathsf{notWEC}}(S,B,a) \wedge \bigcirc \Box B \wedge \bigwedge_{b \in B} \Box \Diamond b).$$

By an abuse of notation, $\Pi_{\mathsf{notWEC}}(S,B,a)$ denotes also an atomic proposition, which holds in a state of the monitored system iff the vector formed by the counters of $\mathcal{M}_{\mathtt{live}}$ is in $\Pi_{\mathsf{notWEC}}(S,B,a)$. As for the minimal elements used in monitoring safety, the sets $\Pi_{\mathsf{notWEC}}(S,B,a)$ cannot be computed in general. In Section 7, we give a class of specifications for which they can. For each $B \subseteq \mathbb{M} \triangleright \mathbb{D}$ (resp., $b \in \mathbb{M} \triangleright \mathbb{D}$), B (resp., b) is an atomic proposition which holds in a state iff the second part of the state of $\mathcal{M}_{\mathtt{live}}$ is in B (resp., is b). Also, $\Diamond$, $\bigcirc$, and $\Box$ denote the temporal operators of LTL eventually, next, and always, respectively.

Theorem 2 (Weak Eventual Consistency Monitoring)**.** *An optimistic replication system $\mathcal{I}$ is weak eventually consistent if and only if $\mathcal{I} \| \mathcal{M}_{\mathtt{safe}}$ cannot reach $q_{\mathtt{err}}$ and $\mathcal{I} \| \mathcal{M}_{\mathtt{live}} \models \neg \varphi_{\mathsf{notWEC}}$.*

6.2 Eventual Consistency

By following the same line of reasoning as the one used for weak eventual consistency, we first derive a necessary and sufficient condition for a trace to be eventually consistent.

In the case of an eventually consistent infinite trace τ, axiom EVENTUAL ensures that, for every finite prefix P of the global interpretation order gi, after some finite prefix of τ, all the operations have P as a prefix of their local interpretations.

If $B \subseteq \mathbb{M} \triangleright \mathbb{D}$ is the set of all a such that τ contains infinitely many a-operations, then the specification of each $a \in B$ must contain an infinite sequence of $\mathbb{M}$-labeled posets such that any prefix P of gi is a prefix of all these posets, except for some finite set. Thus, any prefix P of gi can be extended in order to belong to each of the $S(a)$ with $a \in B$. Moreover, if P contains all the finitely occurring operations in τ, then the extension can only add elements labeled by methods in $\mathsf{meth}(B)$. The set of labeled posets which can be extended in such a way is denoted by $Quot(S,B)$. This implies that an infinite sequence of increasing prefixes of gi must belong to $Quot(S,B)$, which, by extending a classical definition of limit from words to labeled posets, can be stated as the poset defined by gi is in the *limit* of $Quot(S,B)$. Now, since gi is a reordering of τ, this is equivalent to the fact that the multiset of methods that occur in τ is the same as the multiset of methods that occur in some infinite labeled poset belonging to the limit of $Quot(S,B)$. Thus, the eventual consistency of a trace τ can also be characterized in terms of Parikh images. We show in the following that the same monitor $\mathcal{M}_{\mathtt{live}}$ defined previously can be used to reduce the problem of checking eventual consistency to LTL model checking.

Next, we formally define $Quot(S,B)$ and the notion of limit.

Definition 3 (Quotient)**.** Given a specification S, $B \subseteq \mathbb{M} \triangleright \mathbb{D}$, and $a \in B$, let $S(a)\mathsf{meth}(B)^{-1}$ – the *quotient* of S by $\mathsf{meth}(B)$ – be the set of labeled posets for which there exists an $\mathsf{meth}(B)$-completion in $S(a)$. Then, let

$$Quot(S,B) = \bigcap_{a \in B} S(a)\mathsf{meth}(B)^{-1}.$$

Definition 4 (Limit)**.** Given a set $\mathcal{A}$ of finite labeled posets, we denote by $lim(\mathcal{A})$ the set of infinite labeled posets $(A, \leq, \ell)$ which have an infinite sequence of increasing prefixes in $\mathcal{A}$ such that every element in A is in a prefix (and all greater ones).

Remark 1. In the context of totally-ordered sets, the condition that every element in A is in a prefix is already implied by the rest of the definition. However, this is not true in the general case. For instance, let $(A, \leq, \ell)$ be an infinite labeled poset, where $A = \{a_i \mid i \in \mathbb{N}\} \cup \{b_i \mid i \in \mathbb{N}\}$, $\ell(a_i) = a$, $\ell(b_i) = b$ for all $i \geq 0$, $a_0 \leq a_1 \leq \ldots$, and $b_0 \leq b_1 \leq \ldots$. This poset is not in the limit of $\mathcal{A}$, the set of all finite totally-ordered sets where all elements are labeled by a, even though it has an infinite increasing sequence of prefixes which are in $\mathcal{A}$. This is due to the fact that the prefixes don't contain all the elements of $(A, \leq, \ell)$.

This leads us to the following necessary and sufficient condition for eventual consistency.

Lemma 4 (Characterization of Eventual Consistency)**.** *Let $\tau = (O_\tau, po)$ be an infinite in $\tau_p B^\omega$. The trace τ is eventually consistent iff it is safe and $\Pi_{\mathbb{M}}(\tau) \in \Pi_{\mathbb{M}}(lim(Quot(S,B)))$.*

Proof. ($\Rightarrow$) If τ is eventually consistent, then there exist $gi \subseteq O_\tau \times O_\tau$, and for each operation $o \in O_\tau$, $li[o] \subseteq O_\tau \times O_\tau$ satisfying the axioms GIPF, THINAIR, RVAL, EVENTUAL.

It is enough to show that $(\mathbb{O}_\tau, gi, \mathsf{meth}) \in lim(Quot(S,B))$. Let P be a finite prefix of $(\mathbb{O}_\tau, gi, \mathsf{meth})$ containing at least the operations of τ_p. For any $a \in B$, by axiom EVENTUAL, since there are infinitely many a-operations, there exists at least one, o, such that P is a prefix of $li_{\mathbb{M}}[o]$. By RVAL, $li_{\mathbb{M}}[o]$ belongs to $S(a)$, and by the fact that P contains all operations in τ_p, $li_{\mathbb{M}}[o]$ is a $\mathsf{meth}(B)$-completion of P. Thus, $P \in Quot(S,B)$. Since we can find an infinite increasing sequence of such prefixes P containing every operation of O_τ, we have $(\mathbb{O}_\tau, gi, \mathsf{meth}) \in lim(Quot(S,B))$.

($\Leftarrow$) This part of the proof is illustrated on Figure 5. Let $(A, \leq, \ell)$ be an $\mathbb{M}$-labeled poset in $\Pi_{\mathbb{M}}(lim(Quot(S,B)))$ with $\Pi_{\mathbb{M}}(\tau) = \Pi_{\mathbb{M}}(A)$. Define gi such that $(\mathbb{O}_\tau, gi, \mathsf{meth})$ is isomorphic to $(A, \leq, \ell)$ and let f be an isomorphism from A to O_τ.

There exists an infinite sequence of increasing prefixes $A_1, A_2, \ldots$ of A such that every $f(A_i)$ contains the operations of τ_p and such that, for all $a \in B$, and for all $n \in \mathbb{N}^*$, $A_n \in S(a)\mathsf{meth}(B)^{-1}$. Moreover, each operation of O_τ appears in at least one $f(A_i)$, for some i (and in every $f(A_j)$ with $j \geq i$).

Since τ is safe, there exists an issue order e over O_τ that satisfies the conditions in Lemma 1. In the following, we consider that the operations in τ are totally ordered according to e.

The properties on $A_1, A_2, \ldots$ imply that, for every $a \in B$ and every $n \in \mathbb{N}^*$, there exists an $\mathsf{meth}(B)$-completion A_n^a of $f(A_n)$ such that $A_n^a \in S(a)$. We assume that the elements added to $f(A_n)$ to obtain A_n^a come from O_τ and that they occur after the operations in $f(A_n)$. This is possible because τ contains infinitely many a-

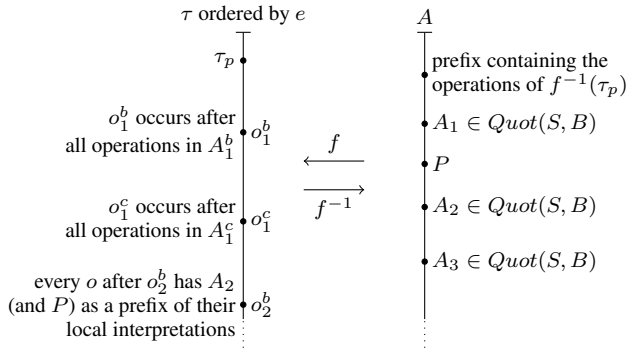


Figure 5: Illustration of the proof of the second part of Lemma 4 for $B = \{b, c\}$

operations, for each $a \in B$. For every $a \in B$ and $n \in \mathbb{N}^*$, let o_n^a be the first operation that occurs in τ after all the operations in A_n^a.

In the following, we define $li[o]$, for every o, such that all the axioms of eventual consistency hold:

- for every $a \in B$, for the finite number of a-operations o, that occur in τ before o_1^a, we use the local interpretations whose existence is ensured by the fact that τ is safe. Similarly, for every $a' \in \mathbb{M} \triangleright \mathbb{D}$ such that τ contains finitely many a'-operations.
- for every $n \in \mathbb{N}^*$, $a \in B$, and for every a-operation o between o_n^a and o_{n+1}^a, we define $li[o] = A_n^a$. Note that this implies that $f(A_n) \preceq li_{\mathbb{M}}[o]$ and that $li_{\mathbb{M}}[o] \in S(a)$.

Axioms THINAIR and RVAL hold for the same reasons given in the proof of Lemma 3. Axiom GIPF holds because of the way we have defined the limit of a set of labeled posets. It remains to show that axiom EVENTUAL holds. Let P be a prefix of (O_τ, gi). Let A_n be one of the previously defined prefixes such that $P \preceq f(A_n)$. For every $a \in B$ and for every a-operation o that occurs after o_n^a, P is a prefix of $li[o]$. Thus, there are only finitely many operations which do not have P as a prefix of their local interpretations, which concludes the proof. □

Monitoring for eventual consistency is similar to the monitoring used for weak eventual consistency. Let $\Pi_{\mathsf{notEC}}(S, B)$ be the set

$$\Pi_{\mathsf{notEC}}(S, B) = \{v \in \mathbb{M} \to \mathbb{N} \mid v + (m \in \mathsf{meth}(B) \mapsto \omega) \notin \Pi_{\mathbb{M}}(lim(Quot(S, B)))\}$$

According to Lemma 4, in order to find a trace which is not eventually consistent, it is enough to look for a trace in $\tau_p B^\omega$ for some τ_p and some $B \subseteq \mathbb{M} \triangleright \mathbb{D}$ such that $\Pi_{\mathbb{M}}(\tau_p) \in \Pi_{\mathsf{notEC}}(S, B)$. This problem can be again reduced to LTL model checking over the parallel composition of $\mathcal{I}$ and the same monitor $\mathcal{M}_{\mathtt{live}}$. In this case, the LTL formula to be checked is:

$$\varphi_{\mathsf{notEC}} = \bigvee_{B \subseteq \mathbb{M} \triangleright \mathbb{D}} \Diamond(\Pi_{\mathsf{notEC}}(S, B) \wedge \bigcirc\Box B \wedge \bigwedge_{b \in B} \Box\Diamond b)$$

As previously, for any $B \subseteq \mathbb{M} \triangleright \mathbb{D}$, $\Pi_{\mathsf{notEC}}(S, B)$ holds in a state of the monitored system if and only if the vector formed by the counters of $\mathcal{M}_{\mathtt{live}}$ is in $\Pi_{\mathsf{notEC}}(S, B)$.

Theorem 3 (Eventual Consistency Monitoring)**.** *An optimistic replication system $\mathcal{I}$ is eventually consistent if and only if $\mathcal{I} \| \mathcal{M}_{\mathtt{safe}}$ cannot reach $q_{\mathtt{err}}$ and $\mathcal{I} \| \mathcal{M}_{\mathtt{live}} \models \neg\varphi_{\mathsf{notEC}}$.*

7. Specifications of finite-state optimistic replication systems

The reductions of eventual consistency to reachability and LTL model checking are effective if one can compute the set of minimal elements in the Parikh image of the specification and effective representations for the sets of vectors $\Pi_{\mathsf{notWEC}}(S, B, a)$ and $\Pi_{\mathsf{notEC}}(S, B)$ defined in Section 6.1 and 6.2. In the following, we introduce automata-based representations for specifications of finite-state optimistic replication systems, for which this is possible.

Essentially, each Σ-labeled poset is abstracted as a sequence of multisets of symbols in Σ which is then recognized by a finite-state automaton where the transitions, instead of being labeled by symbols as in the case of automata over words, are labeled by Presburger constraints. By viewing multisets of symbols as vectors of integers, a sequence of multisets is recognized by an automaton if there exists a run such that the sequence satisfies the constraints imposed by the transitions of this run at each step. The abstraction of a poset as a sequence of multisets is defined based on its decomposition in *levels*, used in algorithms for parallel tasks scheduling [22].

These automata offer a good compromise between simplicity and expressiveness. Since they can recognize words over an alphabet Σ, they can represent specifications, that contain only totally-ordered sets, required by ORS based on the Last Writer Wins conflict resolution policy [15]. They are also able to represent specifications of ORS with commutative conflict resolution policies (i.e., the effect of a set of conflicting operations does not depend on the order in which they are read) such as the CRDTs [7, 20] (see Example 8). For the latter case, the specifications represented by multiset automata are not exactly the ones introduced by the designers of these systems. However, we can prove that eventual consistency w.r.t the original specification is equivalent to eventual consistency w.r.t. the specification recognized by the multiset automaton. In the following, we give a precise statement of this result.

Let $S : \mathbb{M} \triangleright \mathbb{D} \to \mathcal{P}(\mathsf{PoSet}_{\mathbb{M}})$ be a specification and $\sim$ a symmetric binary relation over $\mathbb{M}$, called *commutativity relation*. We say that an $\mathbb{M}$-labeled poset $(A, \leq, \ell)$ is *canonical w.r.t.* $\sim$ iff any two elements labeled by symbols, that are in the relation $\sim$, are incomparable, i.e., for any $x, y \in A$, $\ell(x) \sim \ell(y)$ implies that $x \not\leq y$ and $y \not\leq x$. Then, the specification S is called $\sim$*-closed* iff, for every $a \in \mathbb{M} \triangleright \mathbb{D}$, if $S(a)$ contains a labeled poset $\rho = (A, \leq, \ell)$, then $S(a)$ also contains a labeled poset $\rho' = (A, \leq', \ell)$, which is canonical w.r.t. $\sim$ and such that $\leq' \subseteq \leq$. Furthermore, let $S_\sim$ be a specification s.t. for every $a \in \mathbb{M} \triangleright \mathbb{D}$, $S_\sim(a)$ is the set of posets in $S(a)$, that are canonical w.r.t. $\sim$.

For example, the labeled poset in Figure 1c, belonging to the OR-Set specification, is canonical w.r.t the relation $\sim_O$ that consists of any pair of methods having different arguments (e.g., $\mathtt{add}(i)$ and $\mathtt{rem}(j)$ with $i \neq j$), and any pair formed of an add or a remove and respectively, a lookup (e.g., $\mathtt{add}(i)$ and $\mathtt{lookup}(i)$). The OR-Set specification in Example 6 is $\sim_O$-closed. Also, the MV-Register specification in Example 5 is $\sim_M$-closed, where $\sim_M$ contains any pair of a $\mathtt{wr}(i)$ method and a $\mathtt{rd}$ method.

The following result follows from the fact that eventual consistency imposes rather weak constraints on the local interpretations associated to the operations in a trace.

Proposition 1. *Let $\mathcal{I}$ be an ORS, $\sim$ a symmetric binary relation over $\mathbb{M}$, and S a $\sim$-closed specification. Then, $\mathcal{I}$ is eventually consistent w.r.t S iff $\mathcal{I}$ is eventually consistent w.r.t $S_\sim$.*

A specification S is called *canonical w.r.t.* $\sim$ iff for every $a \in \mathbb{M} \triangleright \mathbb{D}$, $S(a)$ contains only posets, that are canonical w.r.t. $\sim$. Proposition 1 shows that, for ORS like the MV-Register and the OR-Set, it is enough to consider canonical specifications. In the following, we define multiset automata and show how they can be used to represent canonical specifications.

Let $\rho = (A, \leq, \ell)$ be a Σ-labeled poset. The ith level of ρ is the set of elements x in A such that the length of the longest path starting in x is i. A *decomposition* of ρ is a sequence $A_{n-1}, \ldots, A_0$, where $n-1$ is the length of the longest path in ρ and, for all $0 \leq i \leq n-1$, A_i is the ith level of ρ. Note that the decomposition of a labeled poset is unique. The Σ-*decomposition* of ρ is the

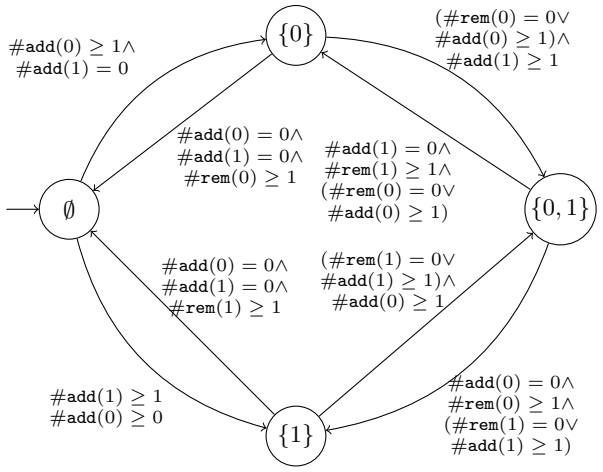


Figure 6: Some transitions of a multiset automaton that recognizes a specification of an OR-Set with at most two elements. The method synopsis $\texttt{lookup}(0) \triangleright 0$ labels the states $\emptyset$ and $\{1\}$, $\texttt{lookup}(0) \triangleright 1$ labels the states $\{0\}$ and $\{0, 1\}$, and so on.

sequence $\Gamma_{n-1}, \ldots, \Gamma_0$, where for all $0 \leq i \leq n-1$, Γ_i is the Parikh image of A_i. By definition, the length of $\Gamma_{n-1}, \ldots, \Gamma_0$ is n.

Example 7. [Σ-decomposition] Consider the $\mathbb{M}_{\mathsf{OR\text{-}S}}$-labeled poset in Figure 1c. The $\mathbb{M}_{\mathsf{OR\text{-}S}}$-decomposition of this labeled poset is:

$$\Gamma_2 = \{\texttt{add}(1), \texttt{rem}(1)\},$$
$$\Gamma_1 = \{\texttt{add}(1), \texttt{rem}(1), \texttt{rem}(1), \texttt{add}(0)\},$$
$$\Gamma_0 = \{\texttt{add}(1), \texttt{rem}(1), \texttt{rem}(0), \texttt{rem}(0)\}.$$

We define an effective representation for isomorphism-closed sets of $\mathbb{M}$-labeled posets by finite automata that recognize their $\mathbb{M}$-decompositions, i.e., sequences of *non-empty* multisets of symbols from $\mathbb{M}$. In order to represent multisets of symbols from $\mathbb{M}$, we consider Presburger formulas φ over a set of free variables $\{\#m \mid m \in \mathbb{M}\}$, where $\#m$ denotes the number of occurrences of the symbol m. Let $\mathcal{F}_{\mathbb{M}}$ denote the set of all such formulas. Also, for any Presburger formula φ, let $[\varphi]$ denote the set of models of φ.

Definition 5 (Multiset Automata)**.** A *multiset automaton* over $\mathbb{M}$ and $\mathbb{M} \triangleright \mathbb{D}$ is a tuple

$$\mathcal{A} = (Q, \delta, Q_0, (Q_a \mid a \in \mathbb{M} \triangleright \mathbb{D})),$$

where Q is a finite set of states, $\delta \subseteq Q \times \mathcal{F}_{\mathbb{M}} \times Q$ is the transition relation, $Q_0 \subseteq Q$ is the set of initial states, and, for any $a \in \mathbb{M} \triangleright \mathbb{D}$, $Q_a \subseteq Q$ (we say that the states in Q_a are labeled by a).

Intuitively, an $\mathbb{M}$-labeled poset is recognized by $\mathcal{A}$ iff there exists a run in $\mathcal{A}$ starting in the initial state such that the transitions are labeled by formulas that are satisfied successively by the Parikh images of the $n-1$th level, $n-2$th level, etc. (where $n-1$ is the length of the longest path). For example, the multiset automaton in Figure 6 recognizes the labeled poset in Figure 1c because there exists a run starting in the initial state that goes through the states labeled by $\emptyset$, $\{1\}$, $\{0, 1\}$, $\{1\}$ such that the associated sequence of formulas describe the $\mathbb{M}_{\mathsf{OR\text{-}S}}$-decomposition in Example 7.

A run of a multiset automaton $\mathcal{A}$ is a sequence $q_0 \xrightarrow{\varphi_0} q_1 \xrightarrow{\varphi_1} \ldots \xrightarrow{\varphi_{n-1}} q_n$, s.t. for every $0 \leq i \leq n-1$, $(q_i, \varphi_i, q_{i+1}) \in \delta$. Such a run *recognizes* an $\mathbb{M}$-labeled poset ρ iff the $\mathbb{M}$-decomposition of ρ is $\Gamma_{n-1}\Gamma_{n-2}\ldots\Gamma_0$ and for every $0 \leq i \leq n-1$, $\Gamma_{n-1-i} \in [\varphi_i]$. We say that the length of this run is n.

Given $q \in Q$, the set of all $\mathbb{M}$-labeled posets recognized by a run of a $\mathcal{A}$, that ends in q, is denoted by $\mathcal{L}(\mathcal{A}, q)$. The labeled posets in $\mathcal{L}(\mathcal{A}, q)$ are said to be *interpreted to* q. Given a set of states $F \subseteq Q$, $\mathcal{L}(\mathcal{A}, F)$ denotes the union of $\mathcal{L}(\mathcal{A}, q)$ with $q \in F$.

Definition 6 (Canonical specifications and multiset automata)**.** A specification S canonical w.r.t $\sim$ is *recognized* by a multiset automaton $\mathcal{A}$ iff for every $a \in \mathbb{M} \triangleright \mathbb{D}$, $S(a)$ is the set of posets in $\mathcal{L}(\mathcal{A}, Q_a)$, that are canonical w.r.t. $\sim$.

Example 8. [A multiset automaton for the OR-Set] Let $S^2_{\mathsf{OR\text{-}S}}$ be a finite-state restriction of the OR-Set specification in Example 6 s.t. (1) the set object contains at most two elements 0 and 1 and (2) $S^2_{\mathsf{OR\text{-}S}}$ contains only posets canonical w.r.t $\sim_O$. This specification is defined over the set of methods `add`, `rem`, and `lookup` with arguments 0 and 1, denoted by $\mathbb{M}^2_{\mathsf{OR\text{-}S}}$. The automaton in Figure 6 recognizes $S^2_{\mathsf{OR\text{-}S}}$.

The following theorem is a direct consequence of the fact that, for any multiset automaton $\mathcal{A}$, one can construct a finite-automaton over sequences whose language has exactly the same Parikh image as the set of labeled posets recognized by $\mathcal{A}$.

Theorem 4. *Let S be a specification recognized by a multiset automaton $\mathcal{A}$. Then, for every $a \in \mathbb{M} \triangleright \mathbb{D}$, there exists an effectively computable Presburger formula ψ_a s.t. $\Pi(S(a)) = [\psi_a]$.*

Theorem 4 implies that, for every $a \in \mathbb{M} \triangleright \mathbb{D}$, the set of minimal elements in $\Pi(S(a))$ is effectively computable and that there exists a computable Presburger formula describing $\Pi_{\mathsf{notWEC}}(S, B, a)$.

Given $B \subseteq \mathbb{M} \triangleright \mathbb{D}$, we show that $\Pi_{\mathsf{notEC}}(S, B)$ defined in Section 6.2 is definable as an effectively computable Presburger formula provided that the specification S is recognized by a multiset automaton $\mathcal{A}$ satisfying some conditions.

First, $\mathcal{A}$ must recognize a prefix-closed set of labeled posets, i.e., $\bigcup_q \mathcal{L}(\mathcal{A}, q)$ is prefix-closed. This is quite natural since the set of all posets in some specification, i.e., $\bigcup_a S(a)$, is usually prefix-closed. Another condition that $\mathcal{A}$ must satisfy can be roughly stated as follows: given a set of methods M and a labeled poset ρ interpreted to some state q, the fact that there exists an M-completion of ρ interpreted to q' depends only on the states q, q', and the set of methods M. Formally, for all $q, q' \in Q$, $M \subseteq \mathbb{M}$, and $\rho_1, \rho_2 \in \mathcal{L}(\mathcal{A}, q)$, $\rho_1 \in \mathcal{L}(\mathcal{A}, q')M^{-1}$ iff $\rho_2 \in \mathcal{L}(\mathcal{A}, q')M^{-1}$. For example, if we consider the automaton in Figure 6, for any labeled poset interpreted to the state labeled by $\{1\}$ there exists an $\{\texttt{add}(0)\}$-completion interpreted to the state labeled by $\{0, 1\}$. In the case of the poset in Figure 1c this completion contains one more element labeled by $\texttt{add}(0)$, which is greater than all the elements labeled by $\texttt{rem}(0)$. Finally, we require that either $\mathcal{A}$ is a *word automaton*, i.e., the transitions are labeled by Presburger formulas that describe singleton multisets, or that for all q, if the limit of $\mathcal{L}(\mathcal{A}, q)$ contains an infinite poset ρ, then it also contains an infinite poset ρ' with the same Parikh image as ρ and such that the decomposition of ρ' has at most $|Q|$ levels. This last condition is satisfied by the automaton in Figure 6 (and an automaton representing an MV-Register specification). Actually, the bound $|Q|$ can be replaced by the constant 2. It is satisfied also by an automaton that describes a specification for the MV-Register. An automaton $\mathcal{A}$ satisfying all these conditions is called *completion-bounded*.

Theorem 5. *Let S be a specification recognized by a completion-bounded multiset automaton $\mathcal{A}$. Then, for every $B \subseteq \Sigma$, there exists an effectively computable Presburger formula φ_B such that $[\varphi_B] = \Pi_{\mathsf{notEC}}(S, B)$.*

Proof sketch. Given $a \in \mathbb{M} \triangleright \mathbb{D}$ and $M \subseteq \mathbb{M}$, we can prove that there exists a maximal set of states $F_{M,a}$ such that $S(a)M^{-1} = \mathcal{L}(\mathcal{A}, F_{M,a})$. To decide if a state q belongs to $F_{M,a}$, one has to consider a minimal labeled poset interpreted to q and search for an M-completion interpreted to a state in Q_a. It can be proved that if there exists such a completion there also exists one of bounded size.

Let $B \subseteq \mathbb{M} \triangleright \mathbb{D}$, $\mathbb{M}_B = \mathsf{meth}(B)$ and $F_B = \bigcap_{a \in B} F_{\mathbb{M}_B, a}$. If $F_B = \emptyset$ then $\varphi_B = true$. Otherwise, we compute a Presburger formula φ that describes the complement of $\Pi_{\mathsf{notEC}}(S, B)$, i.e., the

set of vectors $v \in \mathbb{M} \to \mathbb{N}$ such that $v + (m \in \mathsf{meth}(B) \mapsto \omega) \in \Pi_{\mathbb{M}}(lim(Quot(S,B)))$. By definition, $Quot(S,B) = \mathcal{L}(\mathcal{A}, F_B)$.

First, we prove that $lim(\mathcal{L}(\mathcal{A}, F_B))$ is the union of $lim(\mathcal{L}(\mathcal{A}, q))$ with $q \in F_B$. Let ρ be an infinite labeled poset, which has an infinite set of increasing prefixes $\rho_0, \rho_1, \ldots$ in $\mathcal{L}(\mathcal{A}, F_B)$. Also, let $\theta_0, \theta_1, \ldots$ be an infinite sequence of runs in $\mathcal{A}$ such that, for all i, θ_i is a run that recognizes ρ_i. By Ramsey's theorem, there exist infinitely many runs $\theta_{j_1}, \theta_{j_2}, \ldots$, that end in the same state $q \in F_B$. Thus, the infinitely many posets $\rho_{j_1}, \rho_{j_2}, \ldots$ belong to $\mathcal{L}(\mathcal{A}, q)$, which shows that $\rho \in lim(\mathcal{L}(\mathcal{A}, q))$. Next, we compute for each $q \in F_B$, a formula φ_q that describes the set of vectors $v \in \mathbb{M} \to \mathbb{N}$ such that $v + (m \in \mathsf{meth}(B) \mapsto \omega) \in \Pi_{\mathbb{M}}(lim(\mathcal{L}(\mathcal{A}, q)))$. If $\mathcal{A}$ is a word automaton, then φ_q describes the Parikh image of all the sequences accepted by a run of $\mathcal{A}$ that ends in a cycle on q with at least one transition for each symbol in $\mathbb{M}_B$ and only transitions labeled by symbols in $\mathbb{M}_B$. Thus, φ_q is effectively computable. Otherwise, we enumerate all the runs $q_0 \xrightarrow{\varphi_0} q_1 \xrightarrow{\varphi_1} \ldots \xrightarrow{\varphi_{n-1}} q_n = q$ of $\mathcal{A}$ of length at most $|Q|$. For every such run θ, every index $0 \leq i \leq n-1$, and every surjective mapping $f : \mathbb{M}_B \to \{i, \ldots, n-1\}$, we define a run θ_f as follows. For each $i \leq j \leq n-1$, let φ'_j be a formula describing the set of all multisets v of symbols from $\mathbb{M} \setminus \mathbb{M}_B$ s.t. for every integer L, there exists a multiset modeled by φ_j, that consists of v, at least L symbols m, for all $m \in f^{-1}(j)$, and possibly other symbols from $\mathbb{M}_B$. The run θ_f is obtained from θ by replacing φ_j with $\varphi_j \wedge \varphi'_j$, for all $i \leq j \leq n-1$. The models of φ_q are the Parikh images of all the posets, which are recognized by a run θ_f as above. To prove that the formula φ_q is effectively computable, one can use the same reasoning as in the proof of Theorem 4. Finally, φ is the disjunction of φ_q with $q \in F_B$ and $\varphi_B = \neg\varphi$.

8. Decidability Results

We give decidability results for the case where the specifications are given by multiset automata, and where the optimistic replication system is composed of a fixed number of boolean programs communicating through uni-directional unbounded unordered channels. Each boolean program has instructions $\mathtt{send}(\mathsf{msg}, j)$ that can be used to send a message msg, which belongs to a finite set of messages, to the site identified by j. When a message is sent from a site i to a site j, it is put in an unbounded unordered channel $ch_{i,j}$ from which site j can read by using an instruction $\mathtt{receive}(\mathsf{msg}, i)$. Such systems are called *finite-state optimistic replication systems*.

In order to define an operational model for finite-state optimistic replication systems, we use *Vector Addition Systems with States* (VASSs for short). Formally, a VASS $\mathcal{V}$ is a tuple (Q, d, δ) where Q is a finite set of states, $d \in \mathbb{N}$ is the number of counter variables in the VASS, and $\delta \subseteq Q \times \mathbb{N}^d \times Q$ is the transition relation. The transition system induced by $\mathcal{V}$ is defined in the usual way. A configuration of $\mathcal{V}$ is a pair $(q, \boldsymbol{v})$ where $q \in Q$ and $\boldsymbol{v} \in \mathbb{N}^d$. There is a transition from a configuration $(q, \boldsymbol{v})$ to a configuration $(q', \boldsymbol{v}')$ iff there exists a transition $(q, \boldsymbol{u}, q')$ such that $\boldsymbol{v}' = \boldsymbol{v} + \boldsymbol{u}$.

We briefly describe how to model a finite-state optimistic replication system $\mathcal{I}$ using a VASS $\mathcal{V}$. A configuration of $\mathcal{I}$ is composed of two parts: the first part is a tuple where each component describes the state of a boolean program; the second part is a function, describing the content of each channel. In $\mathcal{V}$, the first part can be encoded in the finite set of states Q. Moreover, if Msg denotes the set of messages exchanged by the sites, the content of an unbounded unordered channel ch of an optimistic replication system can be modeled by $|\mathsf{Msg}|$ counters $ch_1, \ldots, ch_{|\mathsf{Msg}|}$ used to count how many of each kind of messages there are in ch.

We are given a finite-state optimistic replication system $\mathcal{I}$ and a specification S described by a multiset automaton $\mathcal{A}$. As a direct consequence of Theorem 4, given $a \in \mathbb{M} \triangleright \mathbb{D}$, the set of minimal elements in $\Pi(S(a))$ is effectively computable. This implies that also the safety monitor $\mathcal{M}_{\mathtt{safe}}$ can be effectively constructed. Moreover, if $\mathcal{V}$ is a VASS modeling $\mathcal{I}$, we can construct the parallel composition $\mathcal{V} \| \mathcal{M}_{\mathtt{safe}}$ of $\mathcal{V}$ with $\mathcal{M}_{\mathtt{safe}}$. We get a VASS $\mathcal{V}_{\mathtt{safe}}$, on which we can solve the problem of control state reachability, in order to know if it is possible to reach the error state $q_{\mathtt{err}}$ of the monitor. The bound i for the minimal vectors V_a used to construct $\mathcal{M}_{\mathtt{safe}}$ is exponential in $|A|$ and thus, the number of states in $\mathcal{M}_{\mathtt{safe}}$ (and in $\mathcal{V} \| \mathcal{M}_{\mathtt{safe}}$) is also exponential in $|A|$. Moreover solving control-state reachability in VASS is known to be EXPSPACE-complete, which leads to the following theorem.

Theorem 6 (Decidability of Safety)**.** *For a finite-state optimistic replication system $\mathcal{I}$ (given as a* VASS*), and a specification S described by a multiset automaton, the problem of checking the safety of $\mathcal{I}$ w.r.t. S is decidable, and in 2-EXPSPACE.*

The following lemma is used for proving the decidability of both weak eventual consistency and eventual consistency.

Lemma 5. *Let $\mathcal{V} = (Q, d, \delta)$ be a* VASS *where the states are labeled with atomic propositions coming from a finite set AP. Let φ be a Presburger formula with d free variables, and $P \subseteq AP$. The problem of checking the* LTL *formula*

$$\mathcal{V} \models \neg\Diamond(\varphi \wedge \bigcirc\Box P \wedge \bigwedge_{p \in P} \Box\Diamond p)$$

is decidable, and can be reduced to reachability in VASS*.*

Proof. For the formula to be satisfied, there must exist an infinite execution in $\mathcal{V}$ of the form $(q_0, \boldsymbol{v}_0) \ldots (q_i, \boldsymbol{v}_i) \ldots$ where (1) the valuation of the counters in $\boldsymbol{v}_i$ satisfies φ, (2) for all $j > i$, q_j is labeled by a proposition in P, and (3) for each $p \in P$, there are infinitely many q_j labeled by p.

First, Th. 2.14 in Valk and Jantzen [24] shows that it is possible to compute the minimal elements of the upward-closed set of configurations U_P from which there exists an infinite execution visiting only states labeled by a proposition in P and infinitely many states labeled by p, for each $p \in P$. Then, we construct a Presburger formula φ' representing the intersection of $[\varphi]$ and U_P. The problem of checking if there exists a reachable configuration in $\mathcal{V}$ satisfying φ' can be reduced to (configuration) reachability in VASS [4]. □

Theorem 7 (Decidability of (Weak) Eventual Consistency)**.** *Given a finite-state optimistic replication system $\mathcal{I}$, and a specification S described by a multiset automaton, the problem of checking the (weak) eventual consistency of $\mathcal{I}$ w.r.t. S is decidable.*

9. Related Work

Our definition of eventual consistency is inspired by the one given in Burckhardt et al. [6] but it differs from it on several points. In that paper, eventual consistency is defined over traces that are also posets of operations, but the partial order, called *session order*, is defined such that all the operations performed by an user in a session are totally ordered and operations from different sessions are incomparable. Then, a trace is eventually consistent iff there exist two relations over the operations in the trace, called arbitration and resp., visibility relation (denoted by ar and resp., vis), such that (1) the union of vis with the session order is acyclic, (2) for every operation o, the return value of o is associated by the specification to the pair formed of the $\mathbb{M}$-labeled poset $(\mathsf{vis}^{-1}(o), \mathsf{vis}, \mathsf{meth})$ and the projection of ar over the operations in $\mathsf{vis}^{-1}(o)$, (3) ar is a total order, and (4) every operation o is not visible only to some finite set of operations, i.e., for every o, $\{o' \mid (o, o') \notin \mathsf{vis}\}$ is finite.

First, the partial orders over operations that define a trace have different meanings. The operations in a session can be submitted to multiple sites and thus, an infinite extension of the execution in Fig. 2b, where each operation appearing in the figure is executed in a different session, is declared to be eventually consistent although the

union of vis with the session order is acyclic. Then, the definition of a specification in Burckhardt et al. [6] is different because it contains labeled posets augmented with a total order. With our understanding of ORS we think that it is not necessary to add such total orders. A specification should model only the semantics of the operations and some abstraction of the conflict resolution policy. The formalization of the fact that sites will eventually see a consistent state is also different. In fact, the formalization in Burckhardt et al. [6] does not apply to systems that perform speculative executions (this is remarked also by the authors). Intuitively, the order in which two operations are executed cann't change once they are visible. For example, two operations o_1, o_2 with $\mathsf{vis}^{-1}(o_1) = \mathsf{vis}^{-1}(o_2)$, which are instances of the same method, must return the same value. But, for this class of systems, it is possible that the order in which the operations in $\mathsf{vis}^{-1}(o_1)$ are executed changes in between the execution of o_1 and the execution of o_2 and thus, it is possible that the two operations return different values. Another difference is that in our case the local interpretations may converge toward a *partial* order over the operations in the trace, while in Burckhardt et al. [6] they can converge only toward a total order. For instance, this allows that the sites don't have to agree on the order between two conflicting operations, which are commutative.

There are other works that provide formal definitions for eventual consistency, e.g., [3, 9, 20, 21], but they either consider its weaker form, i.e., quiescent consistency, or they use a particular model for the ORS. In general, there exist ORS that guarantee stronger criteria than eventual consistency, e.g., strong eventual consistency [20] or causal+ consistency [17]. Stronger notions for the safety part of eventual consistency, e.g., causal consistency or session guarantees, are formalized in Burckhardt et al. [6] by adding more requirements on the session order and the relation vis. With slight modifications, such requirements can be also added to our definition.

There are works that define decision procedures for the verification of correctness criteria like sequential consistency or linearizability, e.g., [1, 5, 8, 11]. The techniques used in these cases are different from the ones introduced in this paper for eventual consistency, in particular because they are defined over finite traces.

10. Conclusion

We provide an algorithmic approach for verifying eventual consistency of ORS. Our approach is based on reducing the problem of checking eventual consistency to reachability and model-checking problems. This connection is a fundamental result which opens the doors to using existing – exact or approximate – verification tools for message-passing programs (depending on the considered class of systems for the implementation) in the context of verifying ORS.

Our reduction is defined for a general, formally defined, notion of eventual consistency, allowing to reason about a wide class of systems, including those using speculative executions and roll-backs such as Bayou [23] or Telex [2]. In fact, one of the contributions of our paper is to provide such a definition of eventual consistency, as well as a new class of automata that is expressive enough to capture usual specifications of distributed data structures, and which has the needed properties for use in algorithmic verification.

We have in addition shown that when implementations are defined as communicating boolean programs through unbounded unordered channels, the problem of verifying eventual consistency is decidable. As for complexity, our algorithm for checking the safety part of eventual consistency is in 2EXPSPACE, but the problem is EXPSPACE-hard (it is at least as hard as state reachability in VASS). Therefore, an interesting question is whether it is possible to match the two bounds in this case. For the general case of eventual consistency (with liveness), we know that the problem is also EXPSPACE-hard, and that it can be solved using configuration reachability in VASS, which is decidable but with an unknown upper-bound. Then, an interesting question is whether configuration reachability in VASS is needed. We believe that this is the case.

References

[1] R. Alur, K. L. McMillan, and D. Peled. Model-checking of correctness conditions for concurrent objects. *Inf. Comput.*, 160(1-2):167–188, 2000.

[2] L. Benmouffok, J.-M. Busca, J. M. Marquès, M. Shapiro, P. Sutra, and G. Tsoukalas. Telex: A semantic platform for cooperative application development. In *CFSE*, Toulouse, France, 2009.

[3] A.-M. Bosneag and M. Brockmeyer. A formal model for eventual consistency semantics. In *IASTED PDCS*, pages 204–209, 2002.

[4] A. Bouajjani and P. Habermehl. Constrained properties, semilinear systems, and petri nets. In *CONCUR*, pages 481–497, 1996.

[5] A. Bouajjani, M. Emmi, C. Enea, and J. Hamza. Verifying concurrent programs against sequential specifications. In *ESOP*, 2013.

[6] S. Burckhardt, A. Gotsman, and H. Yang. Understanding eventual consistency. Technical Report MSR-TR-2013-39, Microsoft Research.

[7] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: amazon's highly available key-value store. In *SOSP*, 2007.

[8] A. Farzan and P. Madhusudan. Monitoring atomicity in concurrent programs. In *CAV 2008*, pages 52–65. Springer.

[9] A. Fekete, D. Gupta, V. Luchangco, N. A. Lynch, and A. A. Shvartsman. Eventually-serializable data services. In *PODC*, pages 300–309, 1996.

[10] S. Gilbert and N. A. Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33(2):51–59, 2002.

[11] T. A. Henzinger, S. Qadeer, and S. K. Rajamani. Verifying sequential consistency on shared-memory multiprocessor systems. In *CAV*, volume 1633 of *LNCS*, pages 301–315, 1999.

[12] M. Herlihy and N. Shavit. *The art of multiprocessor programming*.

[13] M. Herlihy and J. M. Wing. Linearizability: A correctness condition for concurrent objects. *ACM Trans. Program. Lang. Syst.*, 12(3):463–492, 1990.

[14] J. H. Howard, M. L. Kazar, S. G. Menees, D. A. Nichols, M. Satyanarayanan, R. N. Sidebotham, and M. J. West. Scale and performance in a distributed file system. *ACM Trans. Comput. Syst.*, 6(1):51–81, 1988.

[15] P. R. Johnson and R. H. Thomas. The maintenance of duplicate databases. Technical Report Internet Request for Comments RFC 677, Information Sciences Institute, January 1976.

[16] A.-M. Kermarrec, A. I. T. Rowstron, M. Shapiro, and P. Druschel. The icecube approach to the reconciliation of divergent replicas. In *PODC*, pages 210–218, 2001.

[17] W. Lloyd, M. J. Freedman, M. Kaminsky, and D. G. Andersen. Don't settle for eventual: scalable causal consistency for wide-area storage with COPS. In *SOSP*, pages 401–416, 2011.

[18] J. Michaux, X. Blanc, M. Shapiro, and P. Sutra. A semantically rich approach for collaborative model edition. In *SAC*, pages 1470–1475, 2011.

[19] Y. Saito and M. Shapiro. Optimistic replication. *ACM Comput. Surv.*, 37(1):42–81, 2005.

[20] M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski. A comprehensive study of Convergent and Commutative Replicated Data Types. Rapport de recherche RR-7506, INRIA, Jan. 2011.

[21] M. Shapiro, N. M. Preguiça, C. Baquero, and M. Zawirski. Conflict-free replicated data types. In *SSS*, pages 386–400, 2011.

[22] O. Sinnen. *Task Scheduling for Parallel Systems*. 2007.

[23] D. B. Terry, M. M. Theimer, K. Petersen, A. J. Demers, M. J. Spreitzer, and C. H. Hauser. Managing update conflicts in bayou, a weakly connected replicated storage system. *SIGOPS Oper. Syst. Rev.*, 29(5): 172–182, Dec. 1995.

[24] R. Valk and M. Jantzen. The residue of vector sets with applications to decidability problems in petri nets. *Acta Inf.*, 21:643–674, 1985.