Information Processing Letters 123 (2017) 47–50

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Composing ordered sequential consistency

Kfir Lev-Ari [a,*], Edward Bortnikov [b], Idit Keidar [a,b], Alexander Shraer

[a] Viterbi Department of Electrical Engineering, Technion, Haifa, Israel
[b] Yahoo Research, Haifa, Israel

ARTICLE INFO

Article history:
Received 23 November 2016
Accepted 20 March 2017
Available online 22 March 2017
Communicated by Gregory Chockler

Keywords:
Composability
Consistency
Distributed systems

ABSTRACT

We define *ordered sequential consistency* (OSC), a generic criterion for concurrent objects. We show that OSC encompasses a range of criteria, from sequential consistency to linearizability, and captures the typical behavior of real-world coordination services, such as ZooKeeper. A straightforward composition of OSC objects is not necessarily OSC, e.g., a composition of sequentially consistent objects is not sequentially consistent. We define a global property we call *leading ordered operations*, and prove that it enables correct OSC composition.

1. Introduction

In this work we define a generic correctness criterion named *Ordered Sequential Consistency* (OSC), which captures a range of criteria, from sequential consistency [1] to linearizability [2].

We use OSC to capture the semantics of coordination services such as ZooKeeper [3]. These coordination services provide so-called “strong consistency” for updates and some weaker semantics for reads. They are replicated for high-availability, and each client submits requests to one of the replicas. Reads are not atomic so that they can be served fast, i.e., locally by any of the replicas, whereas update requests are serialized via a quorum-based protocol based on Paxos [4]. Since reads are served locally, they can be somewhat stale but nevertheless represent a valid system state.

In the literature, these services' guarantees are described as atomic writes and FIFO ordered operations for each client [3]. This definition is not tight in two ways: (1) linearizability of updates has no meaning when no operation reads the written values; and (2) this definition allows read operations to read from a future write, which obviously does not occur in any real-world service. A special case of OSC, which we call OSC(U), captures the actual guarantees of existing coordination services.

Although supporting OSC(U) semantics instead of atomicity of all operations enables fast local reads, this makes services *non-composable*: correct OSC(U) coordination services may fail to provide the same level of consistency when combined [5]. Intuitively, the problem arises because OSC(U), similarly to sequential consistency [1], allows subset of operations to occur “in the past”, which can introduce cyclic dependencies.

In a companion systems paper [5] we present ZooNet, a system for modular composition of coordination services, which addresses this challenge: Consistency is achieved on the client side by judiciously adding synchronization requests called *leading ordered operations*. The key idea is to place a “barrier” that limits how far in the past reads can be served from. ZooNet does so by adding a “leading” update request prior to a read request whenever the read is addressed to a different service than the previous one accessed by the same client. We provide here the theoretical underpinnings for the algorithm implemented in ZooNet.

Proving the correctness of ZooNet is made possible by the OSC definition that we present in this paper. Interestingly, Vitenberg and Friedman [6] showed that sequential

* Corresponding author.
E-mail address: kfirla@campus.technion.ac.il (K. Lev-Ari).

http://dx.doi.org/10.1016/j.ipl.2017.03.004
0020-0190/

consistency, when combined with any local (i.e., composable) property continues to be non-composable. Our approach circumvents this impossibility result since having leading ordered operations is not a local property.

2. Model and notation

We use a standard shared memory execution model [2], where a set ϕ of sequential *processes* access shared *objects* from some set X. An object has a name label, a value, and a set of *operations* used for manipulating and reading its value. An operation's execution is delimited by two events, *invoke* and *response*.

A *history* σ is a sequence of operation invoke and response events. An invoke event of operation *op* is denoted i_{op}, and the matching response event is denoted r_{op}. For two events $e_1, e_2 \in \sigma$, we denote $e_1 <_\sigma e_2$ if e_1 precedes e_2 in σ, and $e_1 \leq_\sigma e_2$ if $e_1 = e_2$ or $e_1 <_\sigma e_2$. For two operations op and op' in σ, *op precedes op'*, denoted $op <_\sigma op'$, if $r_{op} <_\sigma i_{op'}$, and $op \leq_\sigma op'$ if $op = op'$ or $op <_\sigma op'$. Two operations are *concurrent* if neither precedes the other.

For a history σ, *complete*(σ) is the sequence obtained by removing all operations with no response events from σ. A history is *sequential* if it begins with an invoke event and consists of an alternating sequence of invoke and response events, s.t. each invoke is followed by the matching response.

For $p \in \phi$, the *process subhistory* $\sigma|p$ of a history σ is the subsequence of σ consisting of events of process p. The *object subhistory* σ_x for an object $x \in$ X is similarly defined. A history σ is *well-formed* if for each process $p \in \phi$, $\sigma|p$ is sequential. For the rest of our discussion, we assume that all histories are well-formed. The order of operations in $\sigma|p$ is called the *process order of p*.

For the sake of our analysis, we assume that each subhistory σ_x starts with a dummy initialization of x that updates it to a dedicated initial value v_0, denoted $di_x(v_0)$, and that there are no concurrent operations with $di_x(v_0)$ in σ_x.

We refer to an operation that changes the object's value as an *update operation*. The *sequential specification* of an object x is a set of allowed sequential histories in which all events are associated with x. For example, the sequential specification of a read-write object is the set of sequential histories in which each read operation returns the value written by the last update operation that precedes it.

3. Ordered sequential consistency

Definition 1 *(OSC(A)).* A history σ is *OSC w.r.t. a subset A of the objects' operations* if there exists a history σ' that can be created by adding zero or more response events to σ, and there is a sequential permutation π of complete(σ'), satisfying the following:

OSC_1 (sequential specification): $\forall x \in$ X, π_x belongs to the sequential specification of x.

OSC_2 (process order): For two operations o and o', if $\exists p \in \phi : o <_{\sigma|p} o'$ then $o <_\pi o'$.

OSC_3 (A-real-time order): $\forall x \in$ X, for an operation $o \in A$ and an operation o' (not necessarily in A) s.t. $o, o' \in \sigma_x$, if $o' <_\sigma o$ then $o' <_\pi o$.

Such π is called a *serialization* of σ. An object is OSC(A) if all of its histories are OSC(A).

We assume that $\forall x \in$ X, $di_x(v_0) \in A$. Linearizability and sequential consistency are both special cases of OSC(A): (1) we get linearizability using A that consist of all of the objects' operations; and (2) we get sequential consistency with A that consists only of dummy initialization operations, which means that there is no operation that precedes an A-operation, i.e., OSC_3 is null, and we left with the sequential specification and process order of an object.

If A consists of the objects' update operations, denoted U, then OSC(U) captures the semantics of coordination services: (1) updates are globally ordered (by OSC_3); and (2) all operations see some prefix of that order (by OSC_3), while respecting each client process order (by OSC_2).

4. OSC(A) composability via leading A-operations

In this section we show that a history σ of OSC(A) objects satisfies OSC(A), if σ has leading ordered A-operations. Generally, we prove the composition by ordering every A-operation o_A on object x, according to the first event $e \in \sigma$ s.t. $e \leq_\sigma r_{o_A}$ and $i_{o_A} <_{\pi_x} e$. Then, we extend that order to a total order on all operations, by placing every non-A-operation after the A-operation that precedes it in their object's serialization. Finally, we show that if σ has leading ordered A-operations, then the total order satisfies OSC(A). Intuitively, we can think of the leading A-operations as a barrier for the non-A-operations, that maintains the total order between objects.

Given a history σ of OSC(A) objects, and a set of serializations $\Pi = \{\pi_x\}_{x \in X}$ of $\{\sigma_x\}_{x \in X}$, we define a strict total order on all operations in Π. We refer to an operation $o \in A$ as an A-operation, and define the future set of an A-operation as follows:

Definition 2 *(A-operation future set).* Given a history σ of OSC(A) objects, an object $x \in \sigma$, a serialization π_x of σ_x, and an A-operation $o_A \in \sigma_x$, the *future set of o_A in π_x* is $F_\sigma^{\pi_x}(o_A) \triangleq \{o \in \pi_x | o_A \leq_{\pi_x} o\}$.

We now define an A-operation's first response event to be the earliest response event of an operation in its future set.

Definition 3 *(First response event).* Given a history σ of OSC(A) objects, an object $x \in \sigma$, a serialization π_x of σ_x, and an A-operation $o_A \in \pi_x$, the *first response event of o_A in π_x*, denoted $fr_\sigma^{\pi_x}(o_A)$, is the earliest response event in σ of an operation in $F_\sigma^{\pi_x}(o_A)$.

Note that it is possible that $fr_\sigma^{\pi_x}(o_A)$ is o_A's response event. We make two observations regarding first responses:

Observation 1. *Given OSC(A) objects' σ, an object $x \in \sigma$, a serialization π_x of σ_x, and an A-operation $o_A \in \pi_x$, then $i_{o_A} <_\sigma fr_\sigma^{\pi_x}(o_A)$.*

Proof. By definition, $fr_\sigma^{\pi_x}(o_A)$ is a response event in σ of an operation o s.t. $o_A \le_{\pi_x} o$. If $fr_\sigma^{\pi_x}(o_A) <_\sigma i_{o_A}$, i.e., $r_o <_\sigma i_{o_A}$, then $o <_\sigma o_A$, a contradiction to OSC_3. □

Observation 2. *Let σ be OSC(A) objects' history, and let π_x be a serialization of σ_x for some x. For two A-operations $o, o' \in \pi_x$, if $o <_{\pi_x} o'$, then $fr_\sigma^{\pi_x}(o) \le_\sigma fr_\sigma^{\pi_x}(o')$.*

Proof. Since $o <_{\pi_x} o'$, we get $F_\sigma^{\pi_x}(o') \subset F_\sigma^{\pi_x}(o)$. By Definition 3, $fr_\sigma^{\pi_u}(o')$ is a response event of an operation $o_1 \in F_\sigma^{\pi_x}(o')$, and therefore $o_1 \in F_\sigma^{\pi_x}(o)$. Thus, $fr_\sigma^{\pi_x}(o)$ is either $fr_\sigma^{\pi_x}(o')$ or an earlier response event in σ. □

To define our strict total order on operations we begin with A-operations:

Definition 4 *(A-Π-order).* Let σ be a history of OSC(A) objects. Let $\Pi = \{\pi_x\}_{x\in X}$ be a set of serializations of $\{\sigma_x\}_{x\in X}$. Let $x, y \in X$, then for two A-operations $o_A \in \pi_x$ and $o'_A \in \pi_y$, we define their A-Π-order, denoted $<_{A\Pi}$, as follows: ($<$) If $x = y$, i.e., $o_A, o'_A \in \pi_x$, then $o_A <_{A\Pi} o'_A$ iff $o_A <_{\pi_x} o'_A$; otherwise, (fr) $x \ne y$, and $o_A <_{A\Pi} o'_A$ iff $fr_\sigma^{\pi_x}(o_A) <_\sigma fr_\sigma^{\pi_y}(o'_A)$.

Lemma 1. *For a history σ of OSC objects and a set of serializations $\Pi = \{\pi_x\}_{x\in X}$ of $\{\sigma_x\}_{x\in X}$, A-Π-order is a strict total order on A-operations in Π.*

Proof. Irreflexivity, antisymmetry, and comparability follow immediately from the definition of $<_{A\Pi}$. We show that $<_{A\Pi}$ satisfies transitivity.

Let o_A, o'_A, and o''_A be three A-operations s.t. $uo_1 <_{A\Pi} uo_2 <_{A\Pi} uo_3$; we need to prove that $uo_1 <_{A\Pi} uo_3$. We consider four cases according to the condition by which each of the pairs is ordered:

($<,<$) If $\exists x \in X$ $o_A, o'_A, o''_A \in \pi_x$, then $o_A <_{\pi_x} o'_A <_{\pi_x} o''_A$ implies $o_A <_{\pi_x} o''_A$, and thus $o_A <_{A\Pi} o''_A$.

($<$, fr) If $\exists x, y \in X, x \ne y : o_A <_{\pi_x} o'_A$, $o''_A \in \pi_y$, and $fr_\sigma^{\pi_x}(o'_A) <_\sigma fr_\sigma^{\pi_y}(o''_A)$, by Observation 2, $fr_\sigma^{\pi_x}(o_A) \le_\sigma fr_\sigma^{\pi_x}(o'_A)$, therefore $fr_\sigma^{\pi_x}(o_A) <_\sigma fr_\sigma^{\pi_y}(o''_A)$, and $o_A <_{A\Pi} o''_A$.

(fr, $<$) If $\exists x, y \in X, x \ne y : o_A \in \pi_x$, $o'_A <_{\pi_y} o''_A$, and $fr_\sigma^{\pi_x}(o_A) <_\sigma fr_\sigma^{\pi_y}(o'_A)$, by Observation 2, $fr_\sigma^{\pi_y}(o'_A) \le_\sigma fr_\sigma^{\pi_y}(o''_A)$. We get $fr_\sigma^{\pi_x}(o_A) <_\sigma fr_\sigma^{\pi_y}(o''_A)$, therefore $o_A <_{A\Pi} o''_A$.

(fr, fr) If $\exists x, y, z \in X, x \ne y, y \ne z : o_A \in \pi_x$, $o'_A \in \pi_y$, and $o''_A \in \pi_z$, this means that $fr_\sigma^{\pi_x}(o_A) <_\sigma fr_\sigma^{\pi_y}(o'_A)$ and $fr_\sigma^{\pi_y}(o'_A) <_\sigma fr_\sigma^{\pi_z}(o''_A)$. By transitivity of $<_\sigma$, $fr_\sigma^{\pi_x}(o_A) <_\sigma fr_\sigma^{\pi_z}(o''_A)$. If $z \ne x$, then $o_A <_{A\Pi} o''_A$. If $z = x$, by the contrapositive of Observation 2, $o_A <_{\pi_x} o''_A$, and $o_A <_{A\Pi} o''_A$. □

We extend $<_{A\Pi}$ to a weak total order in the usual way: $o_1 \le_{A\Pi} o_2$ if $o_1 <_{A\Pi} o_2$ or $o_1 = o_2$. For a history σ, a serialization π_x of σ_x, and an operation o in π_x, the *last A-operation before o in π_x*, denoted $lA_{\pi_x}(o)$, is the latest A-operation in the prefix of π_x that ends with o. Note that if o is an A-operation then $lA_{\pi_x}(o) = o$; and that since every history starts with a dummy initialization, every operation that is not in A is preceded by at least one A-operation and so $lA_{\pi_x}(o)$ is well-defined. We use last A-operations to extend the A-Π-order to a strict total order on all operations in Π.

Definition 5 *(Π-order).* Let σ be a history of OSC(A) objects. Let $\Pi = \{\pi_x\}_{x\in X}$ be a set of serializations of $\{\sigma_x\}_{x\in X}$, and let x and y be objects in X. For two operations $o_1 \in \pi_x$, and $o_2 \in \pi_y$, we define Π-order, denoted $<_\Pi$, as follows: ($lA_{\pi_x}(o_1) \ne lA_{\pi_y}(o_2)$) if the last A-operation before o_1 and o_2 are different, then $o_1 <_\Pi o_2$ iff $lA_{\pi_x}(o_1) <_{A\Pi} lA_{\pi_y}(o_2)$; ($lA_{\pi_x}(o_1) = lA_{\pi_y}(o_2)$) otherwise, $x = y$, and $o_1 <_\Pi o_2$ iff $o_1 <_{\pi_x} o_2$.

We now observe that $<_\Pi$ generalizes all the serializations $\pi_x \in \Pi$:

Observation 3. *Let σ be a history of OSC(A) objects, and $\pi_x \in \Pi$ a serialization of σ_x for some object $x \in X$. For two operations $o_1, o_2 \in \pi_x$, if $o_1 <_{\pi_x} o_2$ then $o_1 <_\Pi o_2$.*

Proof. Since $o_1 <_{\pi_x} o_2$, then $lA_{\pi_x}(o_1) \le_{\pi_x} lA_{\pi_x}(o_2)$. If $lA_{\pi_x}(o_1) = lA_{\pi_x}(o_2)$ then by Definition 5, $o_1 <_\Pi o_2$. Otherwise, by Definition 4, $lA_{\pi_x}(o_1) <_{A\Pi} lA_{\pi_x}(o_2)$ and by Definition 5, $o_1 <_\Pi o_2$. □

Lemma 2. *Let σ be a history of OSC(A) objects, and $\Pi = \{\pi_x\}_{x\in X}$ be a set of serializations of $\{\sigma_x\}_{x\in X}$, then Π-order is a strict total order on all operations in Π.*

Proof. Irreflexivity, antisymmetry, and comparability follow immediately from the definition of $<_\Pi$. We show that $<_\Pi$ satisfies transitivity.

Let o_1, o_2, and o_3 be three operations on objects x, y, z, resp., s.t. $o_1 <_\Pi o_2 <_\Pi o_3$; we need to prove that $o_1 <_\Pi o_3$.

For every o_i and o_j, by Definition 5, $o_i <_\Pi o_j$ implies $lA_{\pi_i}(o_i) \le_{A\Pi} lA_{\pi_j}(o_j)$. By transitivity of $\le_{A\Pi}$ (Lemma 1), we get from $lA_{\pi_x}(o_1) \le_{A\Pi} lA_{\pi_y}(o_2) \le_{A\Pi} lA_{\pi_z}(o_3)$ that $lA_{\pi_x}(o_1) \le_{A\Pi} lA_{\pi_z}(o_3)$.

If $lA_{\pi_x}(o_1) <_{A\Pi} lA_{\pi_z}(o_3)$ then by Definition 5 $o_1 <_\Pi o_3$. If $lA_{\pi_x}(o_1) = lA_{\pi_z}(o_3)$, then by $lA_{\pi_x}(o_1) \le_{A\Pi} lA_{\pi_y}(o_2) \le_{A\Pi} lA_{\pi_z}(o_3)$ we get $lA_{\pi_x}(o_1) = lA_{\pi_y}(o_2) = lA_{\pi_z}(o_3)$, and $x = y = z$. Therefore by $o_1 <_\Pi o_2 <_\Pi o_3$ and Definition 5, $o_1 <_{\pi_x} o_2 <_{\pi_x} o_3$, and thus by Definition 5 $o_1 <_\Pi o_3$. □

Note that Π-order is always defined for compositions of OSC objects. Since it generalizes all the serializations π_x (Observation 3), it preserves OSC_1 and OSC_3. Nevertheless, OSC_2 is not guaranteed.

To support OSC(A) composition we extend each object with a *sync* operation, which does not change the object's state and does not return any value, but belongs to A. For example, to compose OSC($\{di_x(v_0)|\forall x \in X\}$) objects, we extend each of them to be an OSC($\{\text{sync}\} \cup \{di_x(v_0)|\forall x \in X\}$) object and then compose them via adding sync operations.

We say that in a history σ there are *leading ordered operations* if for every operation $o \notin A$ by a process p in σ, the last operation of p before o is on the same object. This also means that between every two operations $o \notin A$ and $o' \notin A$ of different objects by the same process in σ, there is an operation $o_A \in A$ to the second object. We next prove that adding leading ordered operations allows for correct OSC composition.

Theorem 1. *If a history σ of OSC(A) objects has leading ordered operations, then σ is OSC(A).*

Proof. Let $\Pi = \{\pi_x\}_{x \in X}$ be a set of serializations of $\{\sigma_x\}_{x \in X}$, and let π be the sequential permutation of σ defined by $<_\Pi$. We now prove that π satisfies OSC(A). $\mathbf{OSC}_1$ and $\mathbf{OSC}_3$ follow immediately from Observation 3.

We prove $\mathbf{OSC}_2$. Let o_1 and o_2 be two operations in Π for which $\exists p \in \phi : o_1 <_{\sigma|p} o_2$. We now show that $o_1 <_\Pi o_2$.

We start by proving the claim for two consecutive operations in $\sigma|p$. If both operations are on the same object, then by Observation 3, $o_1 <_\Pi o_2$, as needed. Otherwise, $\exists x, y \in X, x \neq y : o_1 \in \pi_x, o_2 \in \pi_y$, and o_1 immediately precedes o_2 in $\sigma|p$. By leading ordered operations, since o_1 and o_2 are not on the same object, o_2 is a A-operation and hence $lA_{\pi_y}(o_2) = o_2$.

By definition, $fr_\sigma^{\pi_x}(lA_{\pi_x}(o_1)) \leq_\sigma r_{o_1}$. Since $r_{o_1} <_\sigma i_{o_2}$, and by Observation 1, $i_{o_2} <_\sigma fr_\sigma^{\pi_y}(o_2)$, we get that $fr_\sigma^{\pi_x}(lA_{\pi_x}(o_1)) <_\sigma fr_\sigma^{\pi_y}(o_2)$. By Definition 4, $lA_{\pi_x}(o_1) <_{A\Pi} o_2$, and by Definition 5, $o_1 <_\Pi o_2$.

Thus, every two consecutive operations $o^i, o^{i+1} \in \Pi$ that are in $\sigma|p$ satisfy $o^i <_\Pi o^{i+1}$. By Lemma 2, $<_\Pi$ is a strict total order on all operations, and therefore by transitivity, we get $o_1 <_\Pi o_2$. □

Acknowledgements

We thank Alexey Gotsman for helpful comments on an earlier draft. Kfir Lev-Ari is supported in part by the Hasso-Plattner Institute (HPI) Research School. This work was partially supported by the Israeli Ministry of Science.

References

[1] L. Lamport, How to make a multiprocessor computer that correctly executes multiprocess programs, IEEE Trans. Comput. 28 (9) (1979) 690–691.

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

[3] P. Hunt, M. Konar, F.P. Junqueira, B. Reed, Zookeeper: wait-free coordination for Internet-scale systems, in: USENIX ATC'10, 2010.

[4] L. Lamport, The part-time parliament, ACM Trans. Comput. Syst. 16 (2) (1998) 133–169.

[5] K. Lev-Ari, E. Bortnikov, I. Keidar, A. Shraer, Modular composition of coordination services, in: USENIX ATC'16, 2016.

[6] R. Vitenberg, R. Friedman, On the locality of consistency conditions, in: DISC'03, 2003.