Brief Announcement: Semantics of Eventually Consistent Replicated Sets
Annette Bieniusa, Marek Zawirski, Nuno Preguiça, Marc Shapiro, Carlos Baquero, Valter Balegas, Sérgio Duarte

To cite this version:
Annette Bieniusa, Marek Zawirski, Nuno Preguiça, Marc Shapiro, Carlos Baquero, et al.. Brief Announcement: Semantics of Eventually Consistent Replicated Sets. DISC 2012 - 26th International Symposium on Distributed Computing, Oct 2012, Salvador, Bahia, Brazil. pp.441–442, 10.1007/978-3-642-33651-5_48 . hal-00769554

HAL Id: hal-00769554
https://inria.hal.science/hal-00769554
Submitted on 2 Jan 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Brief Announcement: Semantics of Eventually Consistent Replicated Sets

Annette Bieniusa¹, Marek Zawirski¹,², Nuno Preguiça³,¹, Marc Shapiro¹, Carlos Baquero⁴, Valter Balegas³, and Sérgio Duarte³

¹ INRIA/LIP6, Paris, France
² UPMC, Paris, France
³ CITI, Universidade Nova de Lisboa, Portugal
⁴ HASLab, INESC Tec and Universidade do Minho, Portugal

This paper studies the semantics of sets under eventual consistency. The set is a pervasive data type, used either directly or as a component of more complex data types, such as maps or graphs. Eventual consistency of replicated data supports concurrent updates, reduces latency and improves fault tolerance, but forgoes strong consistency (e.g., linearisability). Accordingly, several cloud computing platforms implement eventually-consistent replicated sets [2,4].

The sequential semantics of a set are well known, and are defined by individual updates, e.g., \{true\}add\(\{e\}\)\{post-condition\} (in “{pre-condition} computation {post-condition}” notation), where \(S\) denotes its abstract state. However, the semantics of concurrent modifications is left underspecified or implementation-driven.

We propose the following Principle of Permutation Equivalence to express that concurrent behaviour conforms to the sequential specification: “If all sequential permutations of updates lead to equivalent states, then it should also hold that concurrent executions of the updates lead to equivalent states.” It implies the following behavior, for some updates \(u\) and \(u'\):

\[
\{P\}u; u'\{Q\} \land \{P\}u'; u\{Q'\} \land Q \Leftrightarrow Q' \Rightarrow \{P\}u \| u'\{Q\}
\]

Specifically for replicated sets, the Principle of Permutation Equivalence requires that \(\{e \neq f\}\)add\(e\) \| remove\(f\)\{e \in S \And f \notin S\}, and similarly for operations on different elements or idempotent operations. Only the pair add\(e\) \| remove\(e\) is unspecified by the principle, since add\(e\); remove\(e\) differs from remove\(e\); add\(e\). Any of the following post-conditions ensures a deterministic result:

\[
\{\bot_{e \in S}\} \quad \text{– Error mark}
\{e \in S\} \quad \text{– add wins}
\{e \notin S\} \quad \text{– remove wins}
\{add(e) >_{CLK} remove(e) \Leftrightarrow e \in S\} \quad \text{– Last Writer Wins (LWW)}
\]

where \(<_{CLK}\) compares unique clocks associated with the operations. Note that

* This research is supported in part by ANR project ConcoRDanT (ANR-10-BLAN 0208), by ERDF, COMPETE Programme, by Google European Doctoral Fellowship in Distributed Computing received by Marek Zawirski, and FCT projects #PTDC/EIA-EIA/104022/2008 and #PTDC/EIA-EIA/108963/2008.
not all concurrency semantics can be explained as a sequential permutation; for instance no sequential execution ever results in an error mark.

**A Study of Existing Replicated Set Designs.** In the past, several designs have been proposed for maintaining a replicated set. Most of them violate the Principle of Permutation Equivalence (Fig. 1). For instance, the Amazon Dynamo shopping cart [2] is implemented using a register supporting read and write (assignment) operations, offering the standard sequential semantics. When two writes occur concurrently, the next read returns their union. As noted by the authors themselves, in case of concurrent updates even on unrelated elements, a remove may be undone (Fig. 1(a)).

Sovran et al. and Asian et al. [4,1] propose a set variant, C-Set, where for each element the associated add and remove updates are counted. The element is in the abstraction if their difference is positive. C-Set violates the Principle of Permutation Equivalence (Fig. 1(b)). When delivering the updates to both replicas as sketched, the add and remove counts are equal, i.e., $e$ is not in the abstraction, even though the last update at each replica is add($e$).

Shapiro et al. propose a replicated set design, called OR-Set, [3] that ensures that concurrent add/remove operations commute. Unlike the others, it satisfies the Principle of Permutation Equivalence, as illustrated in Figure 1(c). Hidden unique tokens distinguish between different invocations of add, making it possible to precisely track which add operations are affected by a remove.

**References**