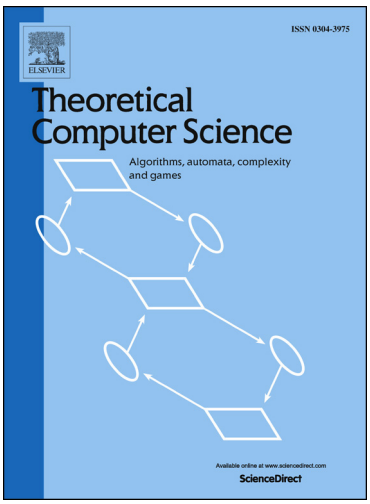
Accepted Manuscript



On the complexity of basic abstractions to implement consensus

Claire Capdevielle, Colette Johnen, Alessia Milani

PII: S0304-3975(18)30005-7
DOI: https://doi.org/10.1016/j.tcs.2017.12.039
Reference: TCS 11442

To appear in: *Theoretical Computer Science*

Received date: 28 March 2017
Revised date: 15 November 2017
Accepted date: 27 December 2017

Please cite this article in press as: C. Capdevielle et al., On the complexity of basic abstractions to implement consensus, *Theoret. Comput. Sci.* (2018), https://doi.org/10.1016/j.tcs.2017.12.039

On the Complexity of Basic Abstractions to Implement Consensus

Claire Capdevielle[a], Colette Johnen[a,*], Alessia Milani[a]

[a] *Univ. Bordeaux, LaBRI, UMR 5800, F-33400 Talence, France*

Abstract

Consensus is one of the central distributed abstractions. By enabling a collection of processes to agree on one of the values they propose, consensus can be used to implement any generic replicated service in a consistent and fault-tolerant way. Therefore, complexity of consensus implementations has become one of the most important topics in the theory of distributed computing. Several concurrent objects have been proposed as building blocks to implement obstruction-free consensus or wait-free consensus in distributed systems augmented with failure detectors or strong synchronization primitives.

In this paper we study an important subset of these objects : adopt-commit [1], conflict-detector [2], value-splitter [3] and grafarius [4]. We show that while some of these objects (adopt-commits and conflict-detectors) ensure a superset of the properties ensured by the others (value-splitter and grafarius), their space and individual step complexity is the same if implemented anonymously (the algorithm does not use process IDs). On the other hand, adopt-commit and conflict-detector objects have a larger complexity if we consider non anonymous implementations.

Keywords: Distributed computing, shared memory, consensus, wait-freedom, complexity, adopt-commit, conflict-detector, value-splitter, grafarius

1. Introduction

Consensus is one of the central abstractions in distributed computing since it can be used to implement any generic replicated service in a consistent and fault-tolerant way. In particular, consensus requires that a collection of processes agree on one of the values they propose.

A fundamental result is that consensus cannot be solved deterministically in an asynchronous read-write shared memory system where a process is guaranteed to decide in a *wait-free* manner (in a finite number of its own steps) [6, 7]. The difficulty stems from handling contended executions. Due to the importance of consensus in dependable distribued computing a lot of work has been devoted to studying how to circumvent this impossibility and to compute the complexity of consensus implementations.

☆The results of section 3 have been presented in [5], this short paper does not contain any proof.

*Corresponding author

Email addresses: claire.capdevielle@labri.fr (Claire Capdevielle), colette.johnen@labri.fr (Colette Johnen), alessia.milani@labri.fr (Alessia Milani)