Brief Announcement: Fully Anonymous Shared Memory Algorithms

Michel Raynal[1,2] and Gadi Taubenfeld[3(✉)]

[1] Univ Rennes IRISA, Rennes, France
[2] Department of Computing, Polytechnic University, Kowloon, Hong Kong
[3] The Interdisciplinary Center, 46150 Herzliya, Israel
tgadi@idc.ac.il

Abstract. Process anonymity has been studied for a long time. Memory anonymity is more recent. In an anonymous memory system, there is no a priori agreement among the processes on the names of the shared registers they access. This article introduces the *fully anonymous* model, namely a model in which both the processes and the memory are anonymous. It is shown that fundamental problems such as mutual exclusion, consensus, and its weak version called set agreement, can be solved despite full anonymity, the first in a failure-free system, the others in the presence of any number of process crashes.

1 Introduction

Process Anonymity. The notion of *process anonymity* has been studied for a long time from an algorithmic and computability point of view, both in message-passing systems and shared memory systems. Process anonymity means that processes have no identity, have the same code and the same initialization of their local variables (otherwise they could be distinguished). We assume a system that is composed of a finite set of $n \geq 2$ asynchronous, anonymous processes denoted $p_1, \ldots, p_n$. Each process knows the number of processes and the number of registers. The subscript i in p_i is only a notational convenience, which is not known by the processes.

Memory Anonymity. The notion of *memory anonymity* has been recently introduced in [8]. Let us consider a shared memory R made up of m atomic registers. Such a memory can be seen as an array with m entries, namely $R[1 \ldots m]$. In a non-anonymous memory system, for each index x, the name $R[x]$ denotes the same register for each process that accesses the address $R[x]$. Hence in a non-anonymous memory, there is an a priori agreement on the names of the shared registers.

The situation is different in an anonymous memory, where there is no a priori agreement on the names of the registers. Moreover, all the registers of an anonymous memory are assumed to be initialized to the same value (otherwise, their initial values could provide information allowing processes to distinguish

M. Ghaffari et al. (Eds.): SSS 2019, LNCS 11914, pp. 301–306, 2019.
https://doi.org/10.1007/978-3-030-34992-9_24

them). The interested reader will find an introductory survey on both process and memory anonymity in [4].

Anonymous Shared Memory. The shared memory is made up of $m \geq 1$ atomic anonymous registers denoted $R[1 \ldots m]$. Hence, *all* the registers are anonymous. As already indicated, due to its anonymity, $R[x]$ does not necessarily indicate the same object for different processes. More precisely, a memory-anonymous system is such that:

- For each process p_i an adversary is defining a permutation $f_i()$ over $\{1, 2, \cdots, m\}$, such that when p_i uses the address $R[x]$, it actually accesses $R[f_i(x)]$,
- No process knows the permutations, and
- All the registers are initialized to the same default value denoted $\perp$.

Table 1. Illustration of an anonymous memory model

Identifiers for an external observer	Local identifiers for process p_i	Local identifiers for process p_j
$R[1]$	$R_i[2]$	$R_j[3]$
$R[2]$	$R_i[3]$	$R_j[1]$
$R[3]$	$R_i[1]$	$R_j[2]$
Permutation	$f_i() : [2, 3, 1]$	$f_j() : [3, 1, 2]$

An example is presented in Table 1. To make apparent the fact that $R[x]$ can have a different meaning for different processes, we write $R_i[x]$ when p_i invokes $R[x]$.

Anonymous Register Models. We consider two types of anonymous register models.

- RW (read/write) model. In this model all, the registers can be atomically read or written by any process.
- RMW (read/modify/write) model. In this model, each register can be atomically read, written or accessed by an operation that atomically reads the register and (according to the value read) possibly modifies it.

Practical Motivation. It was recently shown that epigenetic cell modifications can be modeled by anonymous entities cooperating through anonymous communication media [6]. Hence, fully anonymous distributed systems could inspire bio-informatics (and be inspired by it) [2].

This article considers the following problems.

Mutual Exclusion. Mutual exclusion is the oldest and one of the most important synchronization problems. Formalized by E.W. Dijkstra in the mid-sixties, it consists in building what is called a lock (or mutex) object, defined by two operations, denoted acquire() and release(). For a formal definition, we refer the reader to [3,7].

Consensus. The consensus problem consists in building a one-shot operation, denoted propose(), which takes an input parameter (called *proposed* value) and returns a result (called *decided* value). A process may invoke the operation at most once. The meaning of this operation is defined as follows: (1) Validity: A decided value must be a proposed value; and (2) Agreement: No two processes decide on different values.

The problem must also satisfy one of the following progress conditions: (1) Wait-freedom: If a process does not crash, it must decide; or the weaker (2) Obstruction-freedom: If a process does not crash, and executes alone during a long enough period, it must decide. While wait-free consensus can be solved from registers in a non-anonymous RMW system, it cannot be solved in a non-anonymous RW system. It is possible to solve (the weaker) obstruction-free consensus in non-anonymous RW system.

Set Agreement. Set agreement captures a weaker form of consensus in which the agreement property is weakened as follows: At most $n-1$ different values are decided upon. That is, in any given run, the size of the set of the decision values is at most $n-1$.

The set agreement problem as defined above is also called the $(n-1)$-set agreement problem. While much weaker than consensus, as consensus, wait-free set agreement cannot be solved in non-anonymous RW memory systems.

Content of the Paper. Table 2 describes the technical content of the paper. As an example, the second line states that Sect. 3 presents a consensus algorithm for an anonymous RMW system for $n > 1$ and $m \geq 1$. As far as the mutex algorithm is concerned, it is also shown that $m \in M(n)$, where $M(n) = \{m$ such that $\forall\ \ell :\ 1 < \ell \leq n$: $\mathsf{gcd}(\ell, m) = 1\}$ is a *necessary and sufficient* condition on the size of the memory.

Table 2. Results and the structure of the paper

Problem	Section	Tolerate failures	Register type	Progress condition	Number of processes n	Number of registers m
Mutual exclusion	2	No	RMW	Deadlock-freedom	$n > 1$	$m \in M(n)$
Consensus	3	Yes	RMW	Wait-freedom	$n > 1$	$m \geq 1$
Set agreement	4	Yes	RW	Obstruction-freedom	$n > 1$	$m \geq 3$
Consensus	5	Yes	RW	Obstruction-freedom	$n = 2$	$m \geq 3$

2 Fully Anonymous Mutex Using RMW Registers

The mutual exclusion problem can be solved for non-anonymous processes in both the anonymous RW register model and the anonymous RMW register model [1,8]. However, there is no mutual exclusion algorithm when the processes are anonymous, even when using non-anonymous RW registers. To see that, simply consider an execution in which the anonymous processes run in lock-steps (i.e., one after the other) and access the RW registers in the same order. In such a run it is not possible to break symmetry as the local states of the processes will be exactly the same after each such lock-step.

Let us recall that two integers x and y are said to be *relatively prime* if their greatest common divisor is 1, notice that a number is *not* relatively prime to itself. Let $M(n) = \{m$ such that $\forall \ell : 1 < \ell \leq n : \mathsf{gcd}(\ell, m) = 1\}$.

Theorem 1. *There is a deadlock-free mutual exclusion algorithm for $n \geq 2$ anonymous processes using $m \geq 1$ anonymous* RMW *registers if and only if $m \in M(n)$.*

The proof of the *if direction*, follows from the very existence of the deadlock-free mutual exclusion algorithm for n anonymous processes using m anonymous RMW registers, where $m \in M(n)$, presented in the full version of the paper [5]. The proof of *only if direction*, is a consequence of the lower bound result from [1], which states that $m \in M(n)$ is a necessary and sufficient condition for symmetric deadlock-free mutual exclusion for n non-anonymous processes and m anonymous RMW registers.

3 Fully Anonymous Wait-Free Consensus Using RMW Registers

We describe below a straightforward wait-free consensus algorithm for any number $m \geq 1$ of anonymous registers. This algorithm assumes that the set of proposed values is totally ordered. Each process tries to write the value it proposes into each anonymous register. Assuming that at least one process that does not crash invokes propose(), there is a finite time after which (whatever the concurrency/failure pattern is) each anonymous register contains a proposed value. Then, using the same deterministic rule (for example by choosing the maximum value) the processes decide on the same value.

4 Fully Anonymous Obstruction-Free Set Agreement Using RW Registers

We describe below an obstruction-free set agreement algorithm for $n \geq 2$ anonymous processes using $m \geq 3$ anonymous RW registers. Each anonymous RW register can store the preference of a process. Each participating process scans the m RW registers trying to write its preference into each one of the m registers. Before each write, the process scans the shared array and operates as follows:

- If its preference appears in all the m registers, it reads the array again, and if, for the second time, its preference appears in all the m registers, it decides on its preference and terminates.
- Otherwise, if some preference appears in more than half of the registers, the process adopts this preference as its new preference.

Afterward, the process finds some arbitrary entry in the shared array that does not contain its current preference and writes its current preference into that entry. Once the process finishes writing it repeats the above steps. The exact code and a detailed subtle proof of the algorithm can be found in the full version of this paper [5].

5 Fully Anonymous 2-Process Obstruction-Free Consensus Using RW Registers

As the reader can easily check, instantiating of the obstruction-free set agreement algorithm from the previous section with $n = 2$ provides us with 2-process obstruction-free consensus built using $m \geq 3$ RW registers.

A Conjecture. Let us consider the set agreement algorithm from the previous section, in which it is assumed that $n \geq 2$. We conjecture that when the requirement $m \geq 3$ in this algorithm is strengthened to $m \geq 2n - 1$ the resulting algorithm solves obstruction-free consensus for n processes.

Finally, it was recently proved in [9] that there is no obstruction-free consensus algorithm for two non-anonymous processes using only anonymous bits. Thus, as was shown in [9], anonymous bits are strictly weaker than anonymous (and hence also non-anonymous) multi-valued registers.

6 Discussion

This article has several contributions. The first is the introduction of the notion of *fully anonymous* shared memory systems, namely, systems where the processes are anonymous and there is no global agreement on the names of the shared registers. The article has then addressed the design of mutual exclusion, consensus and set agreement algorithms in specific contexts where the anonymous registers are read/write (RW) registers or more powerful read/modify/write (RMW) registers. It has been shown that, for fully anonymous mutual exclusion using RMW registers, the condition on the number m of registers, namely $m \in M(n)$, is both necessary and sufficient, extending thereby a result of [1] (which was for non-anonymous processes and anonymous registers).
A full version of this paper is available at [5].

Acknowledgments. M. Raynal was partially supported by the French ANR project DESCARTES (16-CE40-0023-03) devoted to layered and modular structures in distributed computing.

References

1. Aghazadeh, Z., Imbs, D., Raynal, M., Taubenfeld, G., Woelfel, Ph.: Optimal memory-anonymous symmetric deadlock-free mutual exclusion. In: Proceedings of 38th ACM Symposium on Principles of Distributed Computing, PODC 2019, 10 p. ACM Press (2019)
2. Navlakha, S., Bar-Joseph, Z.: Distributed information processing in biological and computational systems. Commun. ACM **58**(1), 94–102 (2015)
3. Raynal, M.: Concurrent Programming: Algorithms, Principles and Foundations, 515 p. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-32027-9. ISBN 978-3-642-32026-2
4. Raynal, M., Cao, J.: Anonymity in distributed read/write systems: an introductory survey. In: Podelski, A., Taïani, F. (eds.) NETYS 2018. LNCS, vol. 11028, pp. 122–140. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-05529-5_9
5. Raynal M., Taubenfeld G.: Fully anonymous shared memory algorithms, 16 p. ArXiv-1909.05576 (2019)
6. Rashid S., Taubenfeld G., Bar-Joseph Z.: Genome wide epigenetic modifications as a shared memory consensus problem. In: 6th Workshop on Biological Distributed Algorithms, BDA 2018, London (2018)
7. Taubenfeld, G.: Synchronization Algorithms and Concurrent Programming. Pearson Education/Prentice Hall, London/Upper Saddle River, 423 p. (2006). ISBN 0-131-97259-6
8. Taubenfeld, G.: Coordination without prior agreement. In: Proceedings of 36th ACM Symposium on Principles of Distributed Computing, PODC 2017, pp. 325–334. ACM Press (2017)
9. Taubenfeld, G.: Set agreement power is not a precise characterization for oblivious deterministic anonymous objects. In: Censor-Hillel, K., Flammini, M. (eds.) SIROCCO 2019. LNCS, vol. 11639, pp. 293–308. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-24922-9_20