Can an Operation Both Update the State and Return a Meaningful Value in the Asynchronous PRAM Model?

Jaap-Henk Hoepman

Department of Computer Science, University of Twente, the Netherlands
`hoepman@cs.utwente.nl`

Abstract

On an atomic Read-Modify-Write (RMW) object one can read the complete old contents s of the object and simultaneously update its contents as a function $\delta(s)$ of the old contents in a single, indivisible, atomic operation.

It is known that these RMW objects do not have a wait-free implementation in the asynchronous PRAM model—in which processors can only communicate with each other through atomic read-write registers. For the general case, in which operations P over an object can return a function $\phi_P(s)$ of the old contents s while simultaneously updating the object's state to $\delta_P(s)$, few results are known.

We give several characterisations, in terms of $\phi_P(\cdot)$ and $\delta_P(\cdot)$, of such objects for which no wait-free implementation in the asynchronous PRAM model exists. The resulting objects are remarkably similar to RMW objects. Indeed, we also exhibit two objects satisfying weaker conditions which *do* have a such a wait-free implementation. Our results suggest that only objects as strong as RMW objects do not have wait-free implementation in the asynchronous PRAM model.

Key words: distributed computing, fault tolerance, parallel processing, asynchronous PRAM model, shared memory, read-modify-write objects.

1 Introduction

A Read-Modify-Write (RMW)[KRS86] object is a strong synchronisation primitive that allows one to atomically read and update the contents of a shared memory object. Indeed, their synchronisation properties are so strong that it can be shown that these objects do not have a wait-free implementation in the asynchronous PRAM model [CZ89,Her91].

On the other hand, shared memory objects whose operations either return the current state of the object or update the state of the object, but not both in a single operation, *do* admit wait-free implementation in the asynchronous PRAM model. A prime example is the atomic snapshot object [AAD+93].

In this paper we investigate this property for weaker RMW objects, that lie between these two extremes. For such objects, an operation P can change (part of) the state — modelled by the state transition mapping $\delta_P(\cdot)$ — and return a function of the old contents (possibly masking information) — modelled by the output mapping $\phi_P(\cdot)$. Herlihy has shown [Her91], that if $\delta_P(\cdot)$ is any function unequal to the identity function, while $\phi_P(s) = s$ (i.e., equivalent to a full read), the object does not have a wait-free implementation.

In this paper we extend Herlihy's results by considering also objects with non-trivial output-mappings. We give several characterisations of objects that do not admit a wait-free implementation. Matching our impossibility results we exhibit two objects with slightly weaker (non-trivial) RMW properties that *do* have a wait-free implementation in the asynchronous PRAM model. Our results suggest that only objects almost as strong as true RMW objects do not have a wait-free implementation in the asynchronous PRAM model.

The paper is structured as follows. First, in Section 2 we give a general definition of a shared memory object whose operations can both update the state and return a (meaningful) value, and present the necessary concepts used throughout the paper. Then Section 3 shows two objects that have a wait-free implementation and also satisfy quite strong RMW-like properties. Finally, in section 4 and 5 we derive several characterisations of objects in terms of $\phi_P(\cdot)$ and $\delta_P(\cdot)$ that do not have a wait-free implementation in the asynchronous PRAM model.

2 Definitions

We briefly introduce the necessary concepts and notation in this section. For a more detailed description we refer to [Her91].

All objects $\mathcal{X}$ we consider will have deterministic sequential specifications $\mathbf{S}$. This specifies the set of possible states S of the object, the initial state of the object, and for each operation its effect on the state and its (optional) response when executed atomically. We write $(s, r = O, s') \in \mathbf{S}$ if invoking O in state s changes the state of the object to s' and returns r as its response[1].

[1] In general, operations may have parameters; here it is assumed that for each operation and each of its possible parameters, we have a separate entry in $\mathcal{O}$.

If an operation O does not return a value then we use $r = \perp$.

Alternatively, the behaviour of an operation $O \in \mathcal{O}$ can, in all generality, be expressed by two functions $\delta_O(\cdot)$ — the transformation mapping — and $\phi_O(\cdot)$ — the output mapping — such that for all states $s, s' \in S$, $(s, r = O, s') \in \mathbf{S}$ if and only if $s' = \delta_O(s)$ and $r = \phi_O(s)$.

Histories are finite sequences of operations. For histories H and H', $H \cdot O$ denotes the history obtained by appending operation O after H, $H \cdot H'$ denotes the concatenation of histories H and H', and $\langle H \rangle$ denotes the state which results after applying the operations in H—in order—on the initial state. Thus, $\langle H \cdot O \rangle = \delta_O(\langle H \rangle)$. We will usually omit the triangular brackets, and write $\delta_O(H)$. Let us use the notation $\ell(H)$ for the length of H, and $H[i]$ for the i-th operation in H, for $1 \leq i \leq \ell(H)$. Also let $H\{i : j\}$ be H with the i-th and j-th operation interchanged, while $H[i \cdot\cdot j]$ is the sub-history of H starting with $H[i]$ and ending with $H[j]$. Next we define commuting or overwriting operations.

Definition 1 *Let P and Q be operations of object $\mathcal{X}$. P* commutes with *Q after history H, $P \sim_H Q$, if*

$$\langle H \cdot P \cdot Q \rangle = \langle H \cdot Q \cdot P \rangle \wedge \phi_P(H) = \phi_P(H \cdot Q) \wedge \phi_Q(H \cdot P) = \phi_Q(H) \ .$$

Q overwrites *P after history H, $P \preceq_H Q$, if*

$$\langle H \cdot P \cdot Q \rangle = \langle H \cdot Q \rangle \wedge \phi_Q(H \cdot P) = \phi_Q(H) \ .$$

P commutes with *Q, $P \sim Q$, if for all histories H, $P \sim_H Q$. Similarly, Q* overwrites *P, $P \preceq Q$, if for all histories H, $P \preceq_H Q$.*

We take the following two definitions from Anderson and Moir [AM93] and Aspnes and Herlihy [AH90].

Definition 2 *An object $\mathcal{X}$ is* statically resilient *iff for any two operations P and Q in $\mathcal{O}$ at least one of the following hold: $P \sim Q$, $P \preceq Q$ or $Q \preceq P$.*

Definition 3 *An object $\mathcal{X}$ is* dynamically resilient *iff for every history H and any two operations P and Q in $\mathcal{O}$ at least one of the following hold: $P \sim_H Q$, $P \preceq_H Q$ or $Q \preceq_H P$.*

The second definition allows operations to have a different 'ordering' depending on the history. Clearly, a statically resilient object is also dynamically resilient.

Aspnes and Herlihy [AH90] showed that any dynamically resilient object has an (unbounded) wait-free implementation in the asynchronous PRAM model.

Theorem 4 ([AH90]) *An object has a wait-free implementation using only atomic read/write registers if it is dynamically resilient.*

Matching this, Anderson and Moir [AM93] showed that this condition is necessary for a special class of objects called *snapshot objects*. Next to operations that update the state, these objects also implement a *Read* function that returns the full state of the object. For such a snapshot object to have a wait-free implementation, it must satisfy Definition 3.

We extend their results to a wider class of objects, that do not necessarily implement a full read operation.

3 Weak RMW objects with wait-free implementation

Consider the following two requirements on two operations $A, B \in \mathcal{O}$ for a certain object $\mathcal{X}$.

$$\begin{aligned}(\exists H, H' :: \phi_A(H) \neq \phi_A(H')) \ , \\ (\exists H, H' :: \phi_B(H \cdot H') \neq \phi_B(H \cdot A \cdot H')) \ .\end{aligned} \tag{1}$$

This is in a sense the weakest possible specification of an operation A such that it both updates the state and returns a meaningful value. Indeed, the first part of equation (1) demands that the value returned by A is not a constant, while the second part states that the update of the state by A should be observable through the value returned by some other operation B, perhaps after a certain delay modelled by H'.

Objects satisfying this 'weakest' requirement do sometimes admit a wait-free implementation. One such object is presented in Figure 1. The sequential

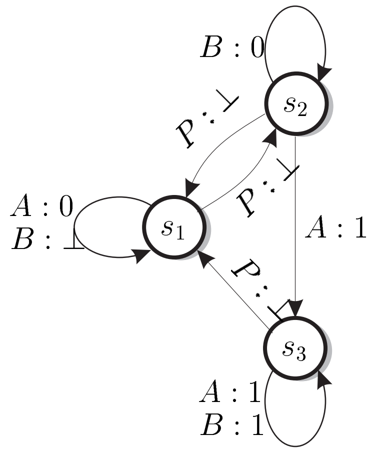

Figure 1. Statically resilient object with weak RMW properties.

specification of the object is represented as a graph where the nodes are all possible states of the object. Edges represent state changes due to the execution of an action. They are labelled by these actions and the value returned by the action. Note that every action is enabled in every state; blocking an action until a precondition holds is not possible for a wait-free object.

Theorem 5 *Object $\mathcal{X}$ of Figure 1 satisfies equation* (1) *and has a wait-free implementation using only atomic read/write registers.*

PROOF. It can be easily checked that this object satisfies equation (1), by observing that $\phi_A(s_1) \neq \phi_A(s_3)$, and $\phi_B(P) \neq \phi_B(P \cdot A)$. Also, $\mathcal{X}$ is statically resilient: $B \preceq A$, $B \preceq P$, $A \preceq P$, $A \preceq A$, $B \sim B$, and $P \sim P$. The result follows by applying Theorem 4. $\square$

A slightly stronger set of requirements is the following.

$$(\exists H_1, H_1' :: \phi_A(H_1 \cdot H_1') \neq \phi_A(H_1 \cdot B \cdot H_1')) \ ,$$
$$(\exists H_2, H_2' :: \phi_B(H_2 \cdot H_2') \neq \phi_B(H_2 \cdot A \cdot H_2')) \ . \tag{2}$$

Here, object $\mathcal{X}$ is required to have two operations A and B that both update the state and return a meaningful value, with the additional constraint that A observes (after a delay H_1') the state change effected by B and vice versa.

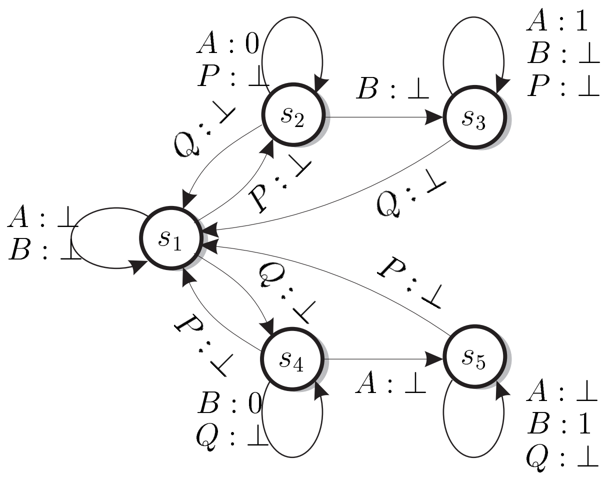

Figure 2. Dynamically resilient object with stronger Scan-and-Update properties.

In the next sections we will show that no statically resilient objects satisfying equation (2) exist. Moreover, objects satisfying equation (2) with the further restriction that $H_1 = H_2$ and both H_1' and H_2' empty do not admit a wait-free implementation In fact, such objects are not dynamically resilient.

However, without further restrictions such a dynamically resilient object admitting a wait-free implementation can be constructed. Indeed, the dynamically resilient object presented in Figure 2 satisfies the even stronger set of requirements

$$\begin{aligned} \phi_A(P) &\neq \phi_A(P \cdot B) , \\ \phi_B(Q) &\neq \phi_B(Q \cdot A) . \end{aligned} \tag{3}$$

Theorem 6 *Object $\mathcal{X}$ of Figure 2 satisfies equation* (3) *(and equation* (2)*) and has a wait-free implementation using only atomic read/write registers.*

PROOF. Equation (3) is easily checked. Clearly equation (2) immediately follows from (3). For each of the states we will establish the 'resilience' order between any pair of operations to show that $\mathcal{X}$ is dynamically resilient.

s_1**:** $A \sim_{s_1} A$, $B \sim_{s_1} B$, $P \sim_{s_1} P$, $Q \sim_{s_1} Q$, $A \sim_{s_1} B$, $A \preceq_{s_1} P$, $A \preceq_{s_1} Q$, $B \preceq_{s_1} P$, $B \preceq_{s_1} Q$, and $P \sim_{s_1} Q$.
s_2**:** $A \sim_{s_2} A$, $B \sim_{s_2} B$, $P \sim_{s_2} P$, $Q \sim_{s_2} Q$, $A \preceq_{s_2} B$, $A \sim_{s_2} P$, $A \preceq_{s_2} Q$, $P \preceq_{s_2} B$, $B \preceq_{s_2} Q$, and $P \preceq_{s_2} Q$.
s_3**:** $A \sim_{s_3} A$, $B \sim_{s_3} B$, $P \sim_{s_3} P$, $Q \sim_{s_3} Q$, $A \sim_{s_3} B$, $A \sim_{s_3} P$, $A \preceq_{s_3} Q$, $B \sim_{s_3} P$, $B \preceq_{s_3} Q$, and $P \sim_{s_3} Q$.
s_4 **and** s_5**:** Symmetric to s_2 and s_3 respectively, with A and B and P and Q interchanged.

Using Theorem 4 this completes the proof. □

4 Impossibility results for dynamically resilient objects

For dynamically resilient objects we have the following straightforward theorem and corollary, that nevertheless give a strong categorisation of those objects that do not have a wait-free implementation. Moreover, in view of Theorem 6, these impossibility results are tight.

The proof of this theorem is similar to Anderson and Moir's proof of the necessity of the dynamic resilience condition [AM93].

Theorem 7 *Consider an object $\mathcal{X}$. If for some operations A and B in $\mathcal{O}$, there exists a history H such that*

$$\phi_A(H) \neq \phi_A(H \cdot B) \quad \textit{and} \quad \phi_B(H) \neq \phi_B(H \cdot A) ,$$

then object $\mathcal{X}$ is not dynamically resilient and does not have a wait-free implementation using only atomic read/write registers.

PROOF. The first part of theorem follows from the fact that if $\phi_A(H) \neq \phi_A(H \cdot B)$ and $\phi_B(H) \neq \phi_B(H \cdot A)$, then neither $A \preceq_H B$ nor $A \sim_H B$ nor $B \preceq_H B$.

The second part of the theorem is proven as follows. Execute the operations in H sequentially, in order, in the implementation of $\mathcal{X}$. This will yield a state $(\langle H \rangle)$ for the implementation of $\mathcal{X}$. Now $\mathcal{X}$, in this state, solves 1-resilient consensus between two processes 0 and 1 using the following protocol

0: **if** A applied to $\mathcal{X}$ returns $\phi_A(H)$
 then decide 0
 else decide 1

1: **if** B applied to $\mathcal{X}$ returns $\phi_B(H)$
 then decide 1
 else decide 0

Loui and Abu-Amara [LAA87] showed that 1-resilient consensus cannot be achieved using only read/write registers. This completes our proof. □

Substituting A for B in the above theorem, we arrive at the following

Corollary 8 *If for object $\mathcal{X}$ there exists an operation $A \in \mathcal{O}$ and a history H such that*

$$\phi_A(H) \neq \phi_A(H \cdot A) ,$$

then $\mathcal{X}$ is not dynamically resilient and does not have a wait-free implementation using only atomic read/write registers.

5 Impossibility results for statically resilient objects

The condition of equation (2) was shown to be implementable by a dynamically resilient object in Theorem 6. In this section we will show that a statically resilient object cannot satisfy equation (2).

We first prove transitivity of the overwrites relation.

Lemma 9 *If $P \preceq_H Q$, $Q \preceq_H R$ and $Q \preceq_{H \cdot P} R$ then $P \preceq_H R$.*

PROOF. By $Q \preceq_{H \cdot P} R$, $\langle H \cdot P \cdot R \rangle = \langle H \cdot P \cdot Q \cdot R \rangle$. By $P \preceq_H Q$, $\langle H \cdot P \cdot Q \cdot R \rangle = \langle H \cdot Q \cdot R \rangle$. By $Q \preceq_H R$, $\langle H \cdot Q \cdot R \rangle = \langle H \cdot R \rangle$. We conclude $\langle H \cdot P \cdot R \rangle = \langle H \cdot R \rangle$.

It remains to show that $\phi_R(H \cdot P) = \phi_R(H)$. By $Q \preceq_H R$ we have $\phi_R(H) = \phi_R(H \cdot Q)$. By $P \preceq_H Q$, $\langle H \cdot P \cdot Q \rangle = \langle H \cdot Q \rangle$ which gives $\phi_R(H) = \phi_R(H \cdot P \cdot Q)$. By $Q \preceq_{H \cdot P} R$ we have $\phi_R(H \cdot P \cdot Q) = \phi_R(H \cdot P)$. $\square$

Corollary 10 *If $P \preceq Q$ and $Q \preceq R$ then $P \preceq R$.*

PROOF. If $P \preceq Q$ and $Q \preceq R$, then for all H, $P \preceq_H Q$, $Q \preceq_H R$ and $Q \preceq_{H \cdot P} R$. Hence for all H, by lemma 9, $P \preceq_H R$. $\square$

Lemma 11 *Suppose $(\exists H, H' :: \phi_Q(H \cdot H') \neq \phi_Q(H \cdot P \cdot H'))$ for a statically resilient $\mathcal{X}$. Then there exists a history H'' such that $\phi_Q(H'') \neq \phi_Q(H'' \cdot P)$.*

PROOF. Pick a pair H, H' satisfying

$$\phi_Q(H \cdot H') \neq \phi_Q(H \cdot P \cdot H') \ , \tag{4}$$

such that H' has minimal length, say k. If $k = 0$ we are done, so we assume $k > 0$.

Proposition 12 *For all i with $1 \leq i < k$ we have $H'[i] \not\preceq H'[i+1]$.*

PROOF. Suppose not. Then for some i we have $\langle H \cdot H' \rangle = \langle H \cdot H'[1 \cdots i-1] \cdot H'[i+1 \cdots k] \rangle$ and $\langle H \cdot P \cdot H' \rangle = \langle H \cdot P \cdot H'[1 \cdots i-1] \cdot H'[i+1 \cdots k] \rangle$. Then H and $H'[1 \cdots i-1] \cdot H'[i+1 \cdots k]$ (with length $k-1$) satisfy Equation ((4)) contradicting the assumption that H' had minimal length. $\square$

Proposition 13 *For all i with $1 \leq i \leq k$ we have $H'[i] \preceq P$.*

PROOF. We prove that for all $R \in H'$ we have $R \preceq P$ by induction over the prefixes of H'. Consider $H'[1 \cdots 1]$. If $H'[1] \sim P$, then $\langle H \cdot P \cdot H' \rangle = \langle H \cdot H'[1] \cdot P \cdot H'[2 \cdots k] \rangle$ and hence $H \cdot H'[1]$ and $H'[2 \cdots k]$ (with length $k-1$) satisfy ((4)) contradicting the assumption that H' had minimal length. If $P \preceq H'[1]$, then $\langle H \cdot P \cdot H' \rangle = \langle H \cdot H' \rangle$ contradicting that $\phi_Q(H \cdot H') \neq \phi_Q(H \cdot P \cdot H')$ Hence by Definition 2, $H'[1] \preceq P$.

Now suppose that for all minimal H' satisfying ((4)) we have $H'[i] \preceq P$. By Proposition 12 and Definition 2 either $H'[i+1] \preceq H'[i]$ or $H'[i] \sim H'[i+1]$. In the first case by Corollary 10 we have $H'[i+1] \preceq P$. In the second case, $\langle H \cdot H' \rangle = \langle H \cdot H'\{i : i+1\} \rangle$ and $\langle H \cdot P \cdot H' \rangle = \langle H \cdot P \cdot H'\{i : i+1\} \rangle$. Hence, H and $H'\{i : i+1\}$ satisfy ((4)), $H'\{i : i+1\}$ has minimal length k too, and $H'[i+1] = H'\{i : i+1\}[i]$. Hence by the induction hypothesis $H'[i+1] = H'\{i : i+1\}[i] \preceq P$. $\square$

Now consider $\langle H \cdot H' \cdot P \rangle$ and $\langle H \cdot P \cdot H' \cdot P \rangle$. There are two cases.

(1) $\phi_Q(H \cdot H' \cdot P) = \phi_Q(H \cdot P \cdot H' \cdot P)$. Using Equation ((4)), either $\phi_Q(H \cdot H') \neq \phi_Q(H \cdot H' \cdot P)$ or $\phi_Q(H \cdot P \cdot H') \neq \phi_Q(H \cdot P \cdot H' \cdot P)$.
(2) $\phi_Q(H \cdot H' \cdot P) \neq \phi_Q(H \cdot P \cdot H' \cdot P)$. Then by Proposition 13 we can eliminate H' and we conclude $\phi_Q(H \cdot P) \neq \phi_Q(H \cdot P \cdot P)$.

This concludes the proof of the lemma. $\square$

Now follow the main two impossibility results of this section

Theorem 14 *A statically resilient object $\mathcal{X}$ with operation $A \in \mathcal{O}$ such that*

$$(\exists H, H' :: \phi_A(H \cdot H') \neq \phi_A(H \cdot A \cdot H'))$$

does not exist.

PROOF. Using Lemma 11 there exists a history H'' such that $\phi_A(H'') \neq \phi_A(H'' \cdot A)$. But then, by Definition 1, neither $A \preceq A$, nor $A \sim A$. This contradicts Definition 2. $\square$

Theorem 15 *A statically resilient object $\mathcal{X}$ with operation $A, B \in \mathcal{O}$ such that*

(1) $(\exists H, H' :: \phi_A(H \cdot H') \neq \phi_A(H \cdot B \cdot H'))$, *and*
(2) $(\exists H, H' :: \phi_B(H \cdot H') \neq \phi_B(H \cdot A \cdot H'))$,

does not exist.

PROOF. Using Lemma 11 there exist histories H and H' such that $\phi_A(H) \neq \phi_A(H \cdot B)$ and $\phi_B(H') \neq \phi_B(H' \cdot A)$. But then, by Definition 1, neither $A \preceq B$, nor $A \sim B$, nor $B \preceq A$. This contradicts Definition 2. $\square$

6 Discussion

We have shown that there exist objects $\mathcal{X}$ with a wait-free implementation using only atomic read/write registers, that satisfy the following weak RMW condition

$$\begin{aligned} &(\exists H_1, H_1' :: \phi_A(H_1 \cdot H_1') \neq \phi_A(H_1 \cdot B \cdot H_1')) \ , \\ &(\exists H_2, H_2' :: \phi_B(H_2 \cdot H_2') \neq \phi_B(H_2 \cdot A \cdot H_2')) \ . \end{aligned} \tag{5}$$

In other words, $\mathcal{X}$ has two operations A and B that both update the state and return a meaningful value, with the additional constraint that in at least one history H_1, A observes (after a delay H_1') the state change effected by B and vice versa (possibly in a different history H_2). Indeed, the delays H_1' and H_2' can be made as short as the empty history.

We show that such an object cannot be statically resilient. If we make equation (5) stronger by requiring that H_1 must be equal to H_2, and keeping $H_1' = H_2' = \varepsilon$, then such an object is not even dynamically resilient and, moreover, does not have a wait-free implementation using atomic read/write registers any more. Notice that RMW objects satisfy this stronger version of equation 5.

It is so far an open question whether we can implement an object using only read/write registers satisfying equation (5) with $H_1' = H_2'$ allowing for a finite delay H_1' and H_2'.

References

[AAD+93] Afek, Y., Attiya, H., Dolev, D., Gafni, E., Merritt, M., and Shavit, N. Atomic snapshots of shared memory. *Journal of the ACM* **40**, 4 (1993), 873–890.

[AM93] Anderson, J. H., and Moir, M. Towards a necessary and sufficient condition for wait-free synchronization. In *7th Int. Workshop on Distributed Algorithms* (Lausanne, Switzerland, 1993), A. Schiper (Ed.), Lect. Not. Comp. Sci. 725, Springer-Verlag, pp. 39–53.

[AH90] Aspnes, J., and Herlihy, M. P. Wait-free data structures in the asynchronous PRAM model. In *2nd Ann. Symp. on Parallel Algorithms and Architectures* (Crete, Greece, 1990), ACM Press, pp. 340–349.

[CZ89] Cole, R., and Zajiec, O. The APRAM: Incorporating asynchrony into the PRAM model. In *Ann. Symp. on Parallel Algorithms and Architectures* (Santa-Fe, NM, 1989), pp. 169–178.

[Her91] Herlihy, M. P. Wait-free synchronization. *ACM Transactions on Programming Languages and Systems* **13**, 1 (1991), 124–149.

[KRS86] Kruskal, C. P., Rudolph, L., and Snir, M. Efficient synchronization on multiprocessors with shared memory. In *5th Ann. Symp. on Principles of Distributed Computing* (Calgary, Alb., Canada, 1986), ACM Press, pp. 218–228.

[LAA87] Loui, M. C., and Abu-Amara, H. H. Memory requirements for agreement among unreliable asynchronous processes. In *Advances in Computing Research*, F. P. Preparata (Ed.), vol. 4. JAI Press, 1987, pp. 163–183.