An Atomicity Criterion for Composite Registers[1]

Lefteris M. Kirousis[2,3] Philippas Tsigas[2,3] and Andreas G. Veneris[2]
E-addresses: kirousis@grpatvx1.bitnet, tsigas@grpatvx1.bitnet,
ven@grpatvx1.bitnet.

Abstract

Let $X_1, \ldots, X_c$ be variables that are shared by a number of processes. These processes are operating in a totally asynchronous and wait-free manner. An operation of a process is either a write on one of the variables or a read of the values of *all* variables. All operations are assumed to be atomic, i.e. an execution of any number of them, including the reads, must be serializable in a way compatible with the values returned by the reads. Such a construction is called an *atomic composite register*. A difficulty usually met in such constructions is to argue about their correctness. In this paper, we give an atomicity criterion for the case of one reader and many writers per variable. The previously known correctness criteria for such constructions depended upon proving the existence of a total order among certain sets of operations on the variables. Exhibiting such a total order has methodologically the same difficulties as showing that the construction is atomic. The criterion presented in this paper does not involve showing the existence of any order, but rather refers to properties of the precedence relation among operations. Showing that these properties are true is a routine job for most known constructions.

[1]This research was partially supported by the ESPRIT II Basic Research Actions Program of the EC under contract no. 3075 (project ALCOM).

[2]Department of Computer Engineering and Informatics, University of Patras, Patras 26110, Greece.

[3]Computer Technology Institute, P.O. Box 1122, Patras 26110, Greece

1 Introduction

An atomic register is essentially a shared variable that can be read or written on by several processes totally asynchronously but in such a way as to *seem* at a higher level that all operations are taking place serially. There are several categories of atomic registers depending upon the number of processes that are allowed to read or write on them or upon the number of bits that can be stored in them. The simplest one is obviously the one-reader, one-writer, one-bit register. It has been shown that using it as a basis more complex ones can be constructed.

The next logical step was the construction of a shared, array-like variable that comprised a number $X_1, \ldots, X_c$ of variables, and was named **composite register**. Each of those X_k , $k = 1, \ldots, c$, shared variables, which we will from now on call **components**, can be written by a number of processes, the **writers**. The values of *all* the components are read in a single atomic operation by one or more processes, the **readers**. The operations of the above processes are assumed to be executed in a totally asynchronous and wait-free manner. A high level operations may have as sub-operations both sub-reads and sub-writes. Recently a number of constructions have been proposed for such atomic registers [Afek et al., 1990], [Anderson, 1990] and subsequently [Kirousis, Spirakis, Tsigas, 1990] and [Kirousis, Spirakis, Tsigas, 1991]. The building blocks of those constructions are atomic, single-component variables, the **sub-registers**. A read or write on the composite register is called *a high-level operation* in order to be distinguished from the *low-level operations* that constitute it and occur in the level of the sub-registers.

The importance of the problem of constructing composite registers is eloquently described in [Afek et al., 1990]: " ...much of the difficulty in proving correctness of concurrent programs is due to the need to argue based on 'inconsistent' views of shared memory, obtained concurrently with other process's modifications. Verification of concurrent algorithms is thus complicated by the need for a 'non-interference' step [...]. By simplifying (or eliminating) the non-interference step, atomic snapshot memories can greatly simplify the design and verification of many concurrent algorithms. "

A formalism for the problem can be found either in [Anderson, 1990] or in [Afek et al., 1990]. In [Lamport, 1986] one can find a formalism for the notion of atomic registers. We tried to keep our notation compatible with these formalisms.

We now proceed to the following definitions:

Definition 1 *A* **reading function** π_k *for a component* X_k *of a composite register is a function that assigns to each high-level read operation* r *a high-level write operation* w *on component* X_k *so that the value returned by* r *for* X_k *is the value written by* w*.*

The way to define such a function at a particular construction is obvious.

Definition 2 *A* **run** *is an execution of an arbitary number of operations according to the respective protocols of the constructions.*

Definition 3 *Let a, b be two distinct high-level operations of the same run. If the last sub-operation of a ends before the start of the first sub-operation of b then we say that a* **precedes** *b (notationally: $a \rightarrow b$). If two operations are incomparable under the precedence relation '$\rightarrow$' we say that they* **overlap**.

Definition 4 *A run ρ is said to be* **atomic** *if and only if there exists a total order $\Rightarrow$ on its high-level operations which is compatible with the precedence relation $\rightarrow$ and which satisfies the property that for each component X_k, $1 \leq k \leq c$, there exists a reading function π_k such that for all high-level read operations r we have that:*

- $\pi_k(r) \Rightarrow r$ *and*
- *not* ($\exists$ write w on component X_k : $\pi_k(r) \Rightarrow w \Rightarrow r$).

Definition 5 *A construction of a composite register is* **correct** *if and only if all of its runs are atomic.*

Intuitively, an atomic construction is one that extends the precedence relation $\rightarrow$, which is a strict partial order on the elements of a run ρ, to a total order $\Rightarrow$ in a way that every high-level read operation will not return a value written after its end or something that has been "overwritten" by another write in the same component.

The above definition, while intuitive, is rather difficult to prove for particular constructions. This is because a total order satisfying the properties in Definition 4 must be defined for every run on the set of high-level operations. Anderson [1990] first proposed an atomicity criterion according to which there is no need to show the existence of a total order among all operations. However, a total order among all writes on each component had still to be defined. Methodologically, this presents almost the same difficulties as showing atomicity directly by the original definition.

In this paper, we give an atomicity criterion for the one-reader, many writers per component case. The advantage of this criterion over previous ones is that it does not entail showing the existence of any kind of order relation among the operations of the construction. It rather refers to properties of the given precedence relation $\rightarrow$, for which, in most cases, it is trivial to prove that they are true.

2 Atomicity Criterion

In this section we give the atomicity criterion.

Theorem 1 *In the case of one reader, many writers per component, c-component composite register, if for a run ρ the following criterion holds :*

not $[(\exists$ *natural* $(k+1): (k+1) \geq 1)$
$(\exists$ *sequence of reads* $r_1, r_2, \ldots, r_k)$
$(\exists$ *reads* $r, r': r_1 \overset{=}{\rightarrow} r$ *and* $r' \overset{=}{\rightarrow} r_k))$
$(\exists$ *naturals* $i_0, i_1, \ldots, i_k: 1 \leq i_j \leq c, \forall j = 1, \ldots, k)$
$(\exists$ *writes* $w_0, w_1, \ldots, w_k$: *each* w_j, $1 \leq j \leq k$, *is a write on component* $i_j)$
$(\pi_{i_1}(r_1) = \pi_{i_1}(r_2)$ *and*
$\pi_{i_2}(r_2) = \pi_{i_2}(r_3)$ *and*
....
$\pi_{i_{k-1}}(r_{k-1}) = \pi_{i_{k-1}}(r_k)$ *and*

$w_0 \overset{=}{\rightarrow} w_1 \overset{=}{\rightarrow} \ldots \overset{=}{\rightarrow} w_k$ *and*
$(\pi_{i_0}(r) \overset{=}{\rightarrow} w_o \neq \pi_{i_0}(r_1)$ *or* $r \rightarrow w_0)$ *and*
$(w_k \overset{=}{\rightarrow} \pi_{i_{k+1}}(r')$ *or* $w_k \rightarrow r'))]$,

where $\overset{=}{\rightarrow}$ *means precedes or equals, then run* ρ *is atomic.*

Proof Assume that the criterion mentioned in the theorem is true for a run ρ. We will show that we can extend the precedence relation $\rightarrow$ to a total order $\Rightarrow$ for all high-level operations of the run, in such a way as for each high level read operation r in run ρ and for each k, where $1 \leq k \leq c$, the value returned by r for the component k will be the same value written by the high-level write operation $\pi_k(r)$ and will not be the case that: $(r \Rightarrow \pi_k(r))$ or $((\exists w)(w$ is a high-level write operation on component $k)(\pi_k(r) \Rightarrow w \Rightarrow r))$.

¿From now on we let r_j to be the jth high level read operation in the linear ordering of the reads, where $1 \leq j \leq$number of reads in run ρ. It is convenient fot the proof to assume, without loss of generality, that there is a high-level read which is preceded by all other high-level operations.

We define the set $S_0(r_n)$ of a read $r_n \in \rho$ to be:

$$S_0(r_n) = \{r_n\} \cup \{\pi_i(r_n): 1 \leq i \leq c\}$$

We define the set $S_1(r_n)$ of a read $r_n \in$ run ρ to be:

$$S_1(r_n) = S_0(r_n) - \bigcup_{r_l \in P(r_n)} S_0(r_l)$$

where $P(r_n)$ is the set defined as follows: $P(r_n) = \{r_l : r_l \rightarrow r_n\}$.

We define the set $W(r_n)$ of a read r_n to be the set of all high level write operations w, such that $\neg((\exists l)(1 \leq l \leq c)(\exists r_j)(\pi_l(r_j) = w))$ and moreover r_n is the least high level read for which if ($\exists$ a sequence of reads $r_{j_1}, r_{j_2}, \ldots, r_{j_k}$) such that:

$[(\exists$ read $r_{j_0} : r_{j_1} \overset{=}{\rightarrow} r_{j_0})$
$(\exists$ natural $i_0, i_1, \ldots, i_k: 1 \leq i_l \leq c, \forall l = 1, \ldots, k)$

$(\exists$ writes $w_0, w_1, \ldots, w_{k-1}$: each w_l is a write on component $i_l)$
$(\pi_{i_1}(r_{j_1}) = \pi_{i_1}(r_{j_2})$ and
$\pi_{i_2}(r_{j_2}) = \pi_{i_2}(r_{j_3})$ and
....
$\pi_{i_{k-1}}(r_{j_{k-1}}) = \pi_{i_{k-1}}(r_{j_k})$ and

$w_0 \stackrel{\equiv}{\rightarrow} w_1 \stackrel{\equiv}{\rightarrow} \ldots \stackrel{\equiv}{\rightarrow} w_{k-1} \stackrel{\equiv}{\rightarrow} w$ and
$(\pi_{i_0}(r_{j_0}) \stackrel{\equiv}{\rightarrow} w_0 \neq \pi_{i_0}(r_{j_1})$ or $r_{j_0} \rightarrow w_0))]$

then $r_{j_k} \rightarrow r_n$. Finally, we define the set $S(r_n)$ of a read r_n to be:

$$S(r_n) = S_1(r_n) \cup W(r_n)$$

Before proceeding with the proof of Theorem 1, we first prove some lemmas. In these lemmas, we assume that the criterion mentioned in Theorem 1 is true for the run ρ.

Lemma 1 *It is not true that there is an l-component write w, $1 \leq l \leq c$, in a $W(r_n)$ for which it is the case that: $\pi_l(r_n) \rightarrow w$.*

Proof Suppose, towards a contradiction, that the required is not true. Then consider a sequence whose only member is r_n and observe that for this sequence and by letting $r_{j_0} = r_{j_1} = r_n$ and $w_0 = w$, we conclude from the way $W(r_n)$ is constructed that $r_{j_1} \rightarrow r_n$, a contradiction.$\Box$

Lemma 2 *It is not true that there is an l-component write w, $1 \leq l \leq c$, in a $W(r_n)$ for which it is the case that: $r_n \rightarrow w$.*

Proof Suppose, towards a contradiction, that the required is not true. Then, as in the previous lemma, consider a sequence whose only member is r_n and observe that for this sequence and by letting $r_{j_0} = r_{j_1} = r_n$ and $w_0 = w$, we conclude from the way $W(r_n)$ is constructed that $r_{j_1} \rightarrow r_n$, a contradiction.$\Box$

Lemma 3 *There does not exist any high-level operation $a \in S(r_l)$ such that :*

$$((\exists r_n)(\exists \text{ write } w)(w \in S(r_n) \text{ and } r_l \rightarrow r_n \text{ and } w \rightarrow a))$$

Proof Suppose, towards a contradiction, that there exists such a high level operation a. We have that some r_{j_k}, $1 \leq j_k <$ number of reads in ρ, directly precedes r_n and therefore $r_l \stackrel{\equiv}{\rightarrow} r_{j_k}$.

We have the following cases:

- $a = r_l$ or $a = \pi_{i_{k+1}}(r_l)$, for some $i_{k+1} : 1 \leq i_{k+1} \leq c$. In this case we obviously violate the criterion, a contradiction.

- $a = w_q$, for some $w_q \in W(r_l)$. If such a w_q exists then, by the definition of $W(r_l)$, we have that there exist $w_0 \stackrel{\equiv}{\rightarrow} w_1 \stackrel{\equiv}{\rightarrow} \ldots \stackrel{\equiv}{\rightarrow} w_{k-1} \stackrel{\equiv}{\rightarrow} w \rightarrow w_q$. But then we have that $r_n \rightarrow r_l$, which is a contradiction.$\Box$

Lemma 4 *There is no high level read operation $r_n \in \rho$ such that:*

$$(\exists l : \ 1 \leq l \leq c \text{ and } r_n \rightarrow \pi_l(r_n))$$

Proof By setting $k = 1$ and $r_1 = r = r'$ and $i_0 = i_1 = i_2$ and $w_1 = w_0 = \pi_{i_2}(r)$ in the criterion, let $i_2 = l$ and $r = r_n$ and so have the following : not $(r_n \rightarrow \pi_l(r_n))$.$\Box$

Now we return to the proof of Theorem 1. We define $a <_j b$ to be a total order on $S(r_j)$, so that the following hold:

- $<_j$ extends $\rightarrow$ restricted to $S(r_j)$.

- In case there is an l-component, $1 \leq l \leq c$, high level write operation w in $W(r_j)$, we have that $w <_j \pi_l(r_j)$ (observe that this condition is compatible with the previous one because of Lemma 1).

- $(\forall a)(a \in S(r_j))$, we have $a <_j r_j$ (again this condition is obviously compatible with the previous ones).

We define:

$a \Rightarrow b$ iff

1. $(a \in S(r_j)$ and $b \in S(r_j)$ and $a <_j b)$ or
2. $(a \in S(r_j)$ and $b \in S(r_i)$ and $r_j \rightarrow r_i)$.

It is clear that the relation $\Rightarrow$ is a total order on the elements of the run ρ defined in such a way that it is not the case that :

- $((\exists r_n)(\exists l)(1 \leq l \leq c)(r_n \Rightarrow \pi_l(r_n)))$ or

- $((\exists l)(1 \leq l \leq c)((\exists w)(w$ is a write on component $l))(\exists r_n)(\pi_l(r_n) \Rightarrow w \Rightarrow r_n))$.

To show that run ρ is atomic, and therefore to complete the proof of the theorem, it only remains to prove that the relation $\Rightarrow$ is an extension of the precedence relation $\rightarrow$. We show this by proving two lemmas.

Lemma 5 *It is not the case that:* $((\exists$ *read* $r_{n_1})(\exists$ *read* $r_{n_2})(\exists l_1)(\exists l_2)$ $(1 \leq l_1, \ l_2 \leq c)(r_{n_1} \rightarrow r_{n_2}$ *and* $\pi_{l_2}(r_{n_1}) \neq \pi_{l_2}(r_{n_2})$ *and* $\pi_{l_2}(r_{n_2}) \rightarrow \pi_{l_1}(r_{n_1})))$.

Proof By setting $k = 1$ and $r' = r_1 \rightarrow r$ in the criterion, let $i_0 = l_1$ and $i_1 = i_2 = l_2$ and $r_{n_1} = r_1$ and $r_{n_2} = r$ and so have the following relation: not $(\pi_{l_1}(r_{n_2}) = w_0 \neq \pi_{l_1}(r_{n_1})$ and $w_1 = \pi_{l_2}(r_{n_1})$ and $w_0 \rightarrow w_1$).$\Box$

Lemma 6 *The relation $\Rightarrow$, which is a total order of the high level operations of the run ρ, is compatible with the precedence relation $\rightarrow$ on the high level operations of the run ρ.*

Proof From Lemma 3 and the way the relation $\Rightarrow$ was defined, it is clear that the way any write w on component l, $\forall l : 1 \leq l \leq c$, such that $\neg((\exists r_n)(\pi_l(r_n) = w))$, relates with whatever other high level operation through the relation $\Rightarrow$ is compatible with the way it relates through the precedence relation. The same is true for the way a $\pi_l(r_n)$, for some $l : 1 \leq l \leq c$, relates with a read r_s. It only remains to prove that if $((\exists \text{read } r_{n_1})(\exists \text{ read } r_{n_2})(r_{n_1} \rightarrow r_{n_2}))$ it is not the case that: $((\exists l_1)(\exists l_2)(1 \leq l_1, \ l_2 \leq c)(\pi_{l_2}(r_{n_1}) \neq \pi_{l_2}(r_{n_2})$ and $\pi_{l_2}(r_{n_2}) \rightarrow \pi_{l_1}(r_{n_1}))$, which has already been proved in the previous Lemma. Therefore, the relation $\Rightarrow$ between all writes w such that $(\exists l)(1 \leq l \leq c)(\exists r_n)(\pi_l(r_n) = w)$ is also compatible with the precedence relation.$\Box$

References

Y. Afek, H. Attiya, D. Dolev, E. Gafni, M. Merritt and N. Shavit (1990): Atomic snapshots of shared memory, *Proceedings of the 9th ACM Symposium on Principles of Distributed Computing, Quebec City, Quebec, Canada.*

J.H. Anderson (1990): Composite registers, *Proceedings of the 9th ACM Symposium on Principles of Distributed Computing, Quebec City, Quebec, Canada.*

L.M. Kirousis, P. Spirakis and Ph. Tsigas (1990): Reading Many Variables in One Atomic Operation: Solutions With Linear Complexity *Technical Report, November 1990, Computer Technology Institute, Patras*

L.M. Kirousis, P. Spirakis and Ph. Tsigas (1991): Simple Atomic Snapshots: A Linear Complexity Solution With Unbounded Time-Stamps, *Proceedings of International Conference on Computing and Information 1991, Ottawa, Canada.*

L. Lamport (1986): On interprocess communication, part i: basic formalism, part ii: basic algorithms, *Distributed Computing* **1**, 77-101.