Synchronizing events in replicated systems

M. Koutny
L. Mancini
Synchronizing events in replicated systems

M. Koutny and L. Mancini

Abstract

The application of N-Modular Redundancy (NMR) in the construction of software permits redundant systems to be robust with respect to failures in replicated processors. In order to exhibit consistent behaviour, all processors in an MNR node must process input requests in the same order. This paper investigates the introduction of synchronization points among replicated computations in a nondeterministic model for concurrent programming. A formal specification is given of a distributed synchronization mechanism that does not introduce any additional deadlock in the replication of a basic single system. We propose this synchronization mechanism as a means of preventing the occurrence of the so-called sequencing failures. The properties of the proposed solution are specified and proved using the COSY path expression formalism. Finally, some implementation schemes are proposed which satisfy the given specification.

To appear in the Journal of Systems and Software.

Series Editor: M.J. Elphick

© 1987 University of Newcastle upon Tyne.
Printed and published by the University of Newcastle upon Tyne, Computing Laboratory, Claremont Tower, Claremont Road, Newcastle upon Tyne, NE1 7RU, England.
**Bibliographical details**

KOUTNY, Maciej


(University of Newcastle upon Tyne, Computing Laboratory, Technical Report Series, no. 237.)

**Added entries**

MANCINI, Luigi Vincenzo

UNIVERSITY OF NEWCASTLE UPON TYNE.


**Abstract**

The application of N-Modular Redundancy (NMR) in the construction of software permits redundant systems to be robust with respect to failures in replicated processors. In order to exhibit consistent behaviour, all processors in an NMR node must process input requests in the same order. This paper investigates the introduction of synchronization points among replicated computations in a nondeterministic model for concurrent programming. A formal specification is given of a distributed synchronization mechanism that does not introduce any additional deadlock in the replication of a basic single system. We propose this synchronization mechanism as a means of preventing the occurrence of the so-called sequencing failures. The properties of the proposed solution are specified and proved using the COSY path expression formalism. Finally, some implementation schemes are proposed which satisfy the given specification.

**About the author**

Dr. M. Koutny has been at the Computing Laboratory as a Research Associate from March, 1985 until Nov. 1986 and as a Lecturer from December, 1986. Prior to this he was at Warsaw Technical University in Poland from 1982.

Mr. L. Mancini has been at the Computing Laboratory since May, 1985 as a Research Associate.

**Suggested keywords**

AGREEMENT
NONDETERMINISM
PATH EXPRESSIONS
REPLICATED PROCESSING
SOFTWARE RELIABILITY

**Suggested classmarks (primary classmark underlined)**

Dewey (18th): 001.6425 001.64404

U.D.C.: 681.322.06 519.687
Synchronizing Events in Replicated Systems

M. Koutny and L. Mancini
Computing Laboratory
The University of Newcastle upon Tyne
Newcastle upon Tyne, NE1 7RU, UK.

ABSTRACT

The application of N-Modular Redundancy (NMR) in the construction of software permits redundant systems to be robust with respect to failures in replicated processors. In order to exhibit consistent behaviour, all processors in an NMR node must process input requests in the same order. This paper investigates the introduction of synchronization points among replicated computations in a nondeterministic model for concurrent programming. A formal specification is given of a distributed synchronization mechanism that does not introduce any additional deadlock in the replication of a basic single system. We propose this synchronization mechanism as a means of preventing the occurrence of the so-called sequencing failures. The properties of the proposed solution are specified and proved using the COSY path expression formalism. Finally, some implementation schemes are proposed which satisfy the given specification.

Index Terms - nondeterminism, agreement, replicated processing, software reliability, path expressions.

1. Introduction

The need to ensure correct input-output behaviour, and a high level of fault-masking in the case of real time systems, has led designers to consider the application of N-Modular Redundancy (N ≥ 2) in the construction of software. Examples of such systems are NASA's Space Shuttle with N = 4 [Skl76], and SRI's SIFT system with N = 3 [Gol80].

This approach to fault-tolerance has demonstrated the possibility of obtaining reliable computations through the replication of programs on multiple computers and the use of a reliable decision algorithm. A decision algorithm may utilize only a subset of all N results for a decision; e.g., the first result that passes an acceptance test may be chosen. Alternatively, it is possible that an acceptable decision result cannot be determined, and a higher level recovery procedure must be invoked. The decision algorithm is often implemented N times - once for each module in which the decision result is used, such as a majority voter in SIFT [Gol80]. In this case, only one computation is affected by the failure of any one implementation.

The many advantages of this approach include: (1) the continuity of correct input/output behaviour in the case of real-time control systems is ensured; (2) reliability is independent of the particular strategy of resource management; (3) transient faults are masked and do not cause reconfiguration.

Modular redundancy in the form of replication of processing modules with majority voting is one of the best known techniques for tolerating failures in concentrated processing. In the present paper we shall explore the application of this form of redundancy to distributed systems composed out of a number of nodes fully connected by means of redundant communication channels. A node consists of N modules in a classical NMR (N-Modular Redundant) configuration. In this paper we specify a fully distributed protocol to allow the establishment of synchronization points among replicated nondeterministic computations. We propose this synchronization mechanism as a means of preventing the occurrence of the so-called sequencing failures [MaSh86] in NMR distributed systems, viz. a failure of the modules within a node to process input requests in the same order.

The COSY (COncurrent SYstem) path expression formalism [LTS79,LSC81] is used to provide a formal specification and correctness proof of our distributed synchronization mechanism. The concurrent model which has been chosen is based on a set of nondeterministic active entities which run in a local protected environment, and interact using message passing only.

The paper is structured into five sections. In Section 2 the NMR correctness requirements are presented. In order to model the synchronization requirements of an NMR system, in Section 3 the basic COSY notation is introduced, and in Section 4 a formal specification of the proposed synchronization mechanism is given. In Section 5, two possible solutions to the problem of
implementing replicated systems are proposed. The first solution is based on the notion of atomic actions. The second solution utilizes a distributed algorithm for cooperation between the system nodes. This paper is a revised and extended version of work reported earlier [MaKo86].

2. Replicated Computation Requirement

The architecture of the system under consideration consists of a number of NMR nodes connected to form an arbitrary graph and communicating only by message passing. Logical connections are established by means of input and output ports. Each NMR node possesses at least one input and one output port.

At a higher level of abstraction a system may be represented as in Fig. 1. Here, circles represent NMR nodes, small white blobs represent output ports, and small black blobs represent input ports. Each pair of corresponding input and output ports (i.e. those which have the same name) is connected by a directed arc which represents a communication channel from output port to input port. One may observe that the input port a is not connected to any other ports of the nodes N1, N2 and N3. This means that a is devoted to external communication, i.e. it can receive requests from the outside environment. Each such port will be called an external port. The whole picture shown in Fig. 1 will be called an NMR-graph.

An NMR-graph models the structure of a system of communicating NMR nodes. The internal structure of an NMR node is shown in Fig. 2, where the NMR node N3 is taken as an example. N3 consists of n (n is fixed) independent modules. The generic module \( M_k \) (1 ≤ k ≤ n) is further composed of a pair of subcomponents \( DA_k \) and \( T_k \) corresponding to the decision algorithm and the task computation performed by module \( M_k \). One can see that the ports of \( N_3 \) are replicated and
assigned to each module $M_k$. It is worth noting that the same input or output port cannot be assigned to the modules of two different $NMR$ nodes, and that the form of communication which is discussed is synchronous and can be considered as one-to-many, i.e. a message sent from an output port, say $p$, is simultaneously and instantaneously received at all input ports $p$.

A correct $NMR$ system must behave like its non-replicated counterpart. We claim that a sufficient condition for this is that every $NMR$ node in the network satisfy the following requirement.

$REQ$: All modules of an $NMR$ node process input requests in an identical order.

It is worth noting that according to requirement $REQ$ all the processors of an $NMR$ node have to resolve nondeterminism (local or global) in an identical manner.

In a distributed system, meeting requirement $REQ$ is a surprisingly difficult task. For example, suppose that a server process receives input requests from two different processes denoted Client$_a$ and Client$_b$, and that to increase reliability it is decided to triplicate the server process.

Assume that Client$_a$ and Client$_b$ both send a service request. Because of communication delays, it is possible, for example, for Server$_1$, Server$_2$, and Server$_3$ to process requests when:

Server$_1$ has received only the service request of Client$_a$;
Server$_2$ has received the service requests of both Client$_a$ and Client$_b$;
Server$_3$ has received only the service request of Client$_b$.

Thus Server$_1$, Server$_2$, and Server$_3$ will generally service requests in different orders and so contravene requirement $REQ$. It is worth noting that a similar situation can occur even in the
absence of communication delays if the client and server processes interact through nondeterministic communication commands.

In order to avoid a diversity of the task computation which will generally result in the sending of inconsistent messages, the modules of an NMR node must synchronize so as to stay in step. In the following a suitable distributed synchronization mechanism will be presented.

3. An Introduction into Basic COSY Syntax and its Standard Semantics

In order to give a formal correctness proof of the proposed protocol, we first introduce the basic syntax and semantics of the COSY formalism [LTS79, LSC81].

COSY is a formalism intended to simplify the study of synchronic aspects of concurrent systems by abstracting away from all aspects of systems except those which have to do with synchronisation.

A basic COSY program or generalized path is a collection of single paths enclosed in program and endprogram parentheses. A single path is a regular expression enclosed by path and end. An example of COSY program is:

```
program path a;b;c end path (d;e)*;b end endprogram
```

In every regular expression like the above, the semicolon denotes sequence (concatenation), and the comma denotes mutually exclusive choice. The comma binds more strongly than semicolon, so that the expression "a;b,c" means "first a, then b or c". An expression may be enclosed in conventional parentheses with a Kleene star appended, as for instance "(d;e)*", which means that the enclosed expression may be executed zero or more times. The expression appearing between path and end is implicitly "starred", so that a single path describes a cyclic sequential subsystem.

A formal description of COSY syntax may be found for instance in [LTS79] and [LSC81]. The semantics of generalized paths can be described by means of vectors of strings [Shi79].

With every single path \( Q = \text{path body end} \), we associate its set of events, \( \text{Ev}(Q) \), which is the set of symbols appearing within body, and its sets of cycles, \( \text{Cyc}(Q) \), which is the regular language generated by body. For example, if \( Q_o = \text{path } (a,b);c \text{ end} \), then \( \text{Ev}(Q_o) = \{a, b, c\} \) and \( \text{Cyc}(Q_o) = \{ac, bc\} \). From the set \( \text{Cyc}(Q) \) we construct the set of firing sequences of \( Q \), denoted by \( \text{FS}(Q) \), as follows:

\[
\text{FS}(Q) = \text{Pref} (\text{Cyc}(Q)^*) = \text{Cyc}(Q)^* \text{Pref} (\text{Cyc}(Q))
\]

where for every alphabet \( A \) and every language \( L \subseteq A^* \), \( \text{Pref}(L) = \{x : \exists y \in A^*, xy \in L\} \). For example \( \text{FS}(Q_o) = (ac, bc)^*\{c, a, ac, b, bc\} \), where \( c \) denotes the empty string.

Let \( P = \text{program } Q_1 \ldots Q_n \text{ endprogram} \) be any generalized path. To model non-sequential behaviour of \( P \), partial orders of occurrences of events will be constructed which are represented by vectors of strings. A vector \( (x_1, \ldots, x_n) \) is a possible behaviour of \( P \) if (1) each \( x_i \) is a possible firing sequence of \( Q_i \), and (2) the \( x_i \)'s agree about the number and order of events the \( Q_i \)'s share. For
example, if \( a, b \in \text{Ev}(Q_1) \cap \text{Ev}(Q_2) \) and \( x_1 = ab \), then \( x_2 \) must be of the form \( x_2 = xyz \), where \( x, y, z \in (\text{Ev}(Q_2) - \text{Ev}(Q_1))^* \). To formally define the set of possible histories of \( P \), vectors of strings are introduced together with a concatenation operation on them.

Let us consider the set \( \text{Ev}(Q_1)^* \times \ldots \times \text{Ev}(Q_n)^* \). If the vectors \((x_1, \ldots, x_n)\) and \((y_1, \ldots, y_n)\) are its members, then their concatenation is defined as: \((x_1, \ldots, x_n) \circ (y_1, \ldots, y_n) = (x_1y_1, \ldots, x_ny_n)\).

Let \( \text{Ev}(P) = \text{Ev}(Q_1) \cup \ldots \cup \text{Ev}(Q_n) \), and let for every \( i, h_i : \text{Ev}(P)^* \rightarrow \text{Ev}(Q_i)^* \) be an erasing homomorphism given by:

\[
\forall a \in \text{Ev}(P), \quad a \in \text{Ev}(Q_i) \Rightarrow h_i(a) = a \\
\forall a \notin \text{Ev}(Q_i) \Rightarrow h_i(a) = e.
\]

For every \( x \in \text{Ev}(P) \), let \( \bar{x} = (h_1(x), \ldots, h_n(x)) \), and let \( \text{Vev}(P) = \{ \bar{x} : a \in \text{Ev}(P) \} \).

The set of all possible histories (behaviours) of \( P \), denoted by \( VFS(P) \), is defined by:

\[
VFS(P) = (\text{FS}(Q_1) \times \ldots \times \text{FS}(Q_n)) \cap \text{Vev}(P)^*.
\]

A generalized path \( P \) is said to be

- deadlock-free if and only if \( \forall x \in VFS(P) \exists a \in \text{Ev}(P), x\bar{x} \in VFS(P) \)
- adequate if and only if \( \forall x \in VFS(P) \forall a \in \text{Ev}(P) \exists z \in \text{Vev}(P)^*, z\bar{a} \in VFS(P) \).

Notice that adequacy is a property akin to absence of partial system deadlock.

4. Formal Specification

This section presents and proves correct formally a COSY specification of a fully distributed mechanism for the synchronization of replicated computation in an NMR system.

The behaviour of a module \( M \) with the internal structure shown in Fig. 3 can be expressed in

![Fig. 3. Internal structure of a module.](image)

the following way: after receiving \( n \) messages at an input port \( p_i \), the decision algorithm \( DA \) produces a single result message which is then processed by the task \( T \). After the completion of the task computation, \( M \) sends result messages to some of its output ports \( q_i \). Therefore, we need to model not only the structure of a system of communicating NMR nodes, but also we have to specify to which ports the result messages (if any) are to be sent. In order to do this we introduce a
mapping \( \phi : In_G \to 2^{Out_G} \), where \( In_G \) and \( Out_G \) denote the set of input and output ports, respectively, of an NMR-graph \( G \), and \( \phi \) satisfies the following two conditions:

1. For every module \( M \), if \( p \in In_M \) then \( \phi(p) \subseteq Out_M \)
2. For every \( q \in Out_G \) there is \( p \in In_G \) such that \( q \in \phi(p) \).

where \( In_M \) and \( Out_M \) denote the input and output ports of a module \( M \). The result of processing \( n \) messages received at input port \( p \) is sent at the set of output ports \( \phi(p) \). In the above, (1) requires that the output ports \( \phi(p) \) belong to the same module as \( p \), while (2) excludes isolated output ports.

A pair \( S = (G, \phi) \) will represent a system of communicating NMR nodes and will be called an NMR-system.

Let us consider an NMR-system \( S = (G, \phi) \). The basic kind of event we will use is:

- \( m_{i,p} \): message sent at output port \( p \) by the \( i \)-th module of the NMR node (if \( p \) is an external input port, then \( m_{i,p} \) denotes the \( i \)-th request received at \( p \)).

Assume that \( M \) is a module, that \( p \in In_M \), and that \( n \) is the number of messages which have to be received at a particular port of \( M \) in order to perform the decision algorithm. To specify \( M \)'s behaviour the following other types of events will be used:

- \( accept_{M,p} \): acceptance test performed by the decision algorithm \( DA \) in \( M \) on \( n \) messages received at input port \( p \);
- \( result_{M,p} \): single decision result (corresponding to \( n \) messages received at \( p \)) sent by \( DA \) to the task \( T \) in \( M \);
- \( set_{M,p} \): the setting of a flag which allows \( M \) to receive the \( n \)-th message at \( p \);
- \( reset_{M,p} \): the releasing of the flag set by \( set_{M,p} \);
- \( task_{M,p} \): computation of task \( T \) related to messages received at \( p \).

Let us suppose that \( M \) is the \( i \)-th module of some NMR node, and that \( In_M = \{p_1, \ldots, p_k\} \), \( Out_M = \{q_1, \ldots, q_m\} \) (as shown in Fig. 3). The behaviour of the module \( M \) is specified by the sequence of single paths:

\[
Module(M) = IN_{M:p_1} \cdots IN_{M:p_k} FLAG_{M:p_1,\ldots,p_k} TASK_{M:p_1,\ldots,p_k} OUT_{M:in(q_1),q_1} \cdots OUT_{M:in(q_m),q_m}
\]

Single paths appearing in \( Module(M) \) are defined in the following way:

\[
IN_{M:p} = \text{path } \Omega_1, p_1; \ldots; \Omega_{n-1}, p_1; set_{M:p_1} (reset_{M:p_1}; set_{M:p})^*; \Omega_{n, p_1} ; accept_{M:p_1} ; result_{M:p_1} ; reset_{M:p_1} \text{ end}
\]

\[
FLAG_{M:p_1,\ldots,p_k} = \text{path } (set_{M:p_1}; \ldots; set_{M:p_k}); (reset_{M:p_1}; \ldots; reset_{M:p_k}) \text{ end}
\]

\[
TASK_{M:p_1,\ldots,p_k} = \text{path } (result_{M:p_1}; task_{M:p_1}; \ldots; (result_{M:p_k}; task_{M:p_k}) \text{ end}
\]
\[ \text{OUT}_{M: \text{in}(q)_q} = \text{path}(\text{task}_{M: u_1}, ..., \text{task}_{M: u_i}); m_{i,q} \text{ end} \]

where \( \Omega_{1,p} = ... = \Omega_{n-1,p} = \Omega_{n,p} = (m_{1,p}, ..., m_{n,p}) \) and \( \text{in}(q) = \{ u_1, ..., u_i \} = \{ r \in \text{In}_M : q \in \phi(r) \} \).

Note that \( \text{TASK}_{M:p_1, ..., p_k} \) and \( \text{FLAG}_{M:p_1, ..., p_k} \) will sometimes be denoted by \( \text{TASK}_M \) and \( \text{FLAG}_M \), respectively.

The informal meaning of the above single paths may be expressed as follows:

\( \text{IN}_{M:p} \) - (Input):

specifies the behaviour of the input port \( p \) associated with \( M \) (after the receipt of \( n \) messages, \( DA \) performs the acceptance test and sends the single result to \( T \));

\( \text{FLAG}_{M:p_1, ..., p_k} \) - (Flags Manager):

allows at most one of \( k \) flags (each one assigned to a different input port \( p_i \)) to be set at any time. This, however, (see the definition of \( \text{IN}_{M:p} \)) prevents the module \( M \) from receiving \( n \) messages at two different input ports at any time;

\( \text{TASK}_{M:p_1, ..., p_k} \) - (Task Computation):

specifies the behaviour of the task \( T \);

\( \text{OUT}_{M: \text{in}(q)_q} \) - (Output):

specifies the behaviour of the output port \( q \) associated with \( M \).

The specification of an NMR node \( N \) is the sequence of single paths

\[ \text{Node}(N) = \text{Module}(M_1)... \text{Module}(M_n) \]

where \( M_i \) is the \( i \)-th module of \( N \), for \( i = 1, ..., n \).

Finally, assuming that \( \{N_1, ..., N_k\} \) is the set of all NMR nodes of \( G \), and that \( \text{In}_G \cup \text{Out}_G = \{p_1, ..., p_m\} \), the formal specification of the NMR-system \( S = (G, \phi) \) is expressed by the following generalized path:

\[ \text{NMRSYST}(S) = \text{program} \text{Node}(N_1)... \text{Node}(N_k) \, \text{Syn}(p_i)... \text{Syn}(p_m) \, \text{endprogram} \]

where the single paths \( \text{Syn}(p) \) express the assumption that a module with input port \( p \) is willing to accept a sequence of \( n \) messages at that port, each message in the sequence being sent by a different module with output port \( p \). For every \( p \in \text{In}_G \cup \text{Out}_G \), \( \text{Syn}(p) \) is defined as follows:

\[ \text{Syn}(p) = \text{path} \, \text{Exp}(m_{1,p}, ..., m_{n,p}) \, \text{end} \]

In the above \( \text{Exp}(m_{1,p}, ..., m_{n,p}) \) is any regular expression written in the COSY syntax which generates the language of all strings \( x_1x_2...x_n \) such that \( x_i \in \{ m_{1,p}, ..., m_{n,p} \} \) and \( x_i \neq x_j \) for \( i \neq j \). For example, \( \text{Exp}(m_{1,p}, m_{2,p}) \) may be defined as \( (m_{1,p}; m_{2,p}); (m_{2,p}; m_{1,p}) \).
Example. Let $S = (G, \phi)$ be an NMR-system ($n = 2$) given by the NMR-graph $G$ shown in Fig. 4,

![Diagram](image)

Fig. 4

and $\phi$ defined as follows:

$\phi(a) = \{b\}, \phi(b) = \{c, d\}, \phi(d) = \{e\}, \phi(e) = \{c\}.$

Moreover, let $A$ and $B$ denote respectively the first and the second module of $N_1$, and let $C$ and $D$ denote respectively the first and the second module of $N_2$. The formal specification of $S$ is given by the following generalized path:

$\text{NMRSYST}(S) =$

```
program
  Syn(a) Syn(b) Syn(c) Syn(d) Syn(e)
  IN_{A,a} IN_{A,d} FLAG_{A,a,d} TASK_{A,a,d} OUT_{A,(a),(b)} OUT_{A,(d),(e)}
  IN_{B,a} IN_{B,d} FLAG_{B,a,d} TASK_{B,a,d} OUT_{B,(a),(b)} OUT_{B,(d),(e)}
  IN_{C,e} IN_{C,e} FLAG_{C,b,e} TASK_{C,b,e} OUT_{C,(b),(e)} OUT_{C,(d),(b,d)}
  IN_{D,b} IN_{D,e} FLAG_{D,b,e} TASK_{D,b,e} OUT_{D,(b),(e)} OUT_{D,(d),(b,d)}
endprogram
```

The specification presented satisfies the fundamental requirement of NMR systems, namely $\text{REQ}$, which was introduced in Section 2. To prove this we need some auxiliary notions.

Let $P = \text{NMRSYST}(S) = \text{program } P_1 \ldots P_m \text{ endprogram}$, and let $\text{Result}$ denote the set of all events of $Ev(P)$ which represent messages sent from the decision algorithms to the task computations, i.e. those which have the form $\text{result}_{M,p}$. For every $x \in VFS(P)$, and for every $i \leq m$, let $\bar{x}P_i = x_i$, where $x = (x_1, \ldots, x_i, \ldots, x_m)$ (i.e. $\bar{x}P_i$ is the projection of $x$ onto subsystem $P_i$).

Finally, let $\Phi : Ev(P)^* \rightarrow \text{In}_G^*$ denote a homomorphism defined as follows:

$\forall a \in Ev(P), [a \notin \text{Result} \Rightarrow \Phi(a) = \varepsilon] \& [a = \text{result}_{M,p} \in \text{Result} \Rightarrow \Phi(a) = p]$
Definition. We say that a history $\mathbf{x} \in VFS(P)$ satisfies \textit{REQ} iff for every pair $M,L$ of modules of every NMR node one of the following two conditions is satisfied:

\begin{enumerate}
\item \(\Phi(\mathbf{x}|\text{TASK}_M) = \Phi(\mathbf{x}|\text{TASK}_L)\)
\item \(\exists p \in \text{In}_G, \Phi(\mathbf{x}|\text{TASK}_M) = \Phi(\mathbf{x}|\text{TASK}_L)p \lor \Phi(\mathbf{x}|\text{TASK}_L) = \Phi(\mathbf{x}|\text{TASK}_M)p\).
\end{enumerate}

That is, if a history satisfies the requirement \textit{REQ}, then either all the $n$ modules of each particular NMR node performed the same sequences of requests or these sequences differ in that there is one request (represented by $p$) which has not yet been performed by a subset of modules, say $\{M_i\}$. In such a case $p$ is the next request to be performed by $\{M_j\}$.

**THEOREM 1.** Each history $\mathbf{x} \in VFS(P)$ satisfies \textit{REQ}.

**PROOF**

Let $\mathbf{x} \in VFS(P)$, and let $M$ and $L$ ($M = L$) be modules of an NMR node.

For every $K \in \{L,M\}$, and for every $p \in \text{In}_K$ we define $\Psi_{K,p}(\mathbf{x})$ as follows:

Let $y = \mathbf{x}[\text{IN}_K,p]$. One may unambiguously decompose $y$ as $y = wz$, where $w$ is the maximal prefix of $y$ which belongs to $Cyc(\text{IN}_K,p)^*$. Then we define $\Psi_{K,p}(\mathbf{x})$ to be the number of occurrences of events representing messages (i.e. $m_1,p,...,m_{n,p}$) within $z$.

Clearly, $0 \leq \Psi_{K,p}(\mathbf{x}) \leq n$. (Intuitively, $\Psi_{K,p}(\mathbf{x})$ is the number of messages waiting for the acceptance test after being received at $p$).

One may observe that the single path \textit{FLAG}_K guarantees that:

\begin{equation}
\forall q \neq p, \Psi_{K,p}(\mathbf{x}) = n \Rightarrow \Psi_{K,q}(\mathbf{x}) < n
\end{equation}

i.e. $K$ cannot receive $n$ messages at two different input ports at any time.

For every alphabet $A$, and for all $b \in A$ and $s \in A^*$, let $\#_b(s)$ denote the number of occurrences of $b$ within $s$.

We have for every $p \in \text{In}_M (= \text{In}_L)$:

\begin{enumerate}
\item \(#_p(\Phi(\mathbf{x}|\text{TASK}_M)) - #_p(\Phi(\mathbf{x}|\text{TASK}_L)) \leq 1\)
\item \(#_p(\Phi(\mathbf{x}|\text{TASK}_M)) - #_p(\Phi(\mathbf{x}|\text{TASK}_L)) = 1 \Rightarrow \Psi_{L,p}(\mathbf{x}) = n\)
\item \(#_p(\Phi(\mathbf{x}|\text{TASK}_L)) - #_p(\Phi(\mathbf{x}|\text{TASK}_M)) = 1 \Rightarrow \Psi_{M,p}(\mathbf{x}) = n\)
\end{enumerate}

Notice that (4.2), (4.3.a) and (4.3.b) follow essentially from the fact that single paths $\text{IN}_M,p$ and $\text{IN}_{L,p}$ share events $m_{1,p},...,m_{n,p}$.
Let us define a mapping \( l : \text{Vev}(P)^* \to \{0,1,2,\ldots\} \) as follows:

1. \( l(\varepsilon) = 0 \)
2. \( \forall x \in \text{Vev}(P)^* \forall a \in \text{Ev}(P), l(xa) = 1 + l(x) \),

where \( x \in \text{Vev}(P)^* \) is the vector the coordinates of which is the empty string.

We prove the theorem by induction on \( l(x) \), the length of \( x \).

Clearly, \( \Phi(x|\text{TASK}_{M}) = c = \Phi(x|\text{TASK}_{L}) \).

Assume now that \( x \in \text{VFS}(P) \) satisfies \( \text{REQ} \), and that \( xa \in \text{VFS}(P), a \in \text{Ev}(P) \). We have to prove that \( xa \) satisfies \( \text{REQ} \).

Clearly, if \( a \notin B = (\text{Result} \cap (\text{Ev}(\text{TASK}_{M}) \cup \text{Ev}(\text{TASK}_{L}))) \), then \( xa \) satisfies \( \text{REQ} \), so let assume \( a \in B \). If \( \Phi(x|\text{TASK}_{M}) = \Phi(x|\text{TASK}_{L}) \), then \( xa \) satisfies \( \text{REQ} \). Otherwise, by the induction hypothesis, we may assume without loss of generality that \( \Phi(x|\text{TASK}_{M}) = \Phi(x|\text{TASK}_{L})p \), where \( p \in \text{In}_{M} \). Hence, by (4.3.a), \( \Psi_{L,p}(x) = n \).

We now consider two cases.

Case 1: \( a = \text{result}_{M,q} \). By (4.2), \( p = q \). Thus, by (4.3.a), \( \Psi_{L,q}(xa) = n \). On the other hand, \( \Psi_{L,p}(xa) = \Psi_{L,p}(x) = n \), which produces a contradiction with (4.1).

Case 2: \( a = \text{result}_{L,q} \). By (4.1), \( \Psi_{L,r}(x) < n \), for every \( r \neq p \). Thus, \( q = p \).

Hence, we have shown that \( a = \text{result}_{L,p} \).

Consequently, \( \Phi(xa|\text{TASK}_{L}) = \Phi(x|\text{TASK}_{L})p = \Phi(xa|\text{TASK}_{M}) \).

Therefore, \( xa \) satisfies \( \text{REQ} \). ☐

Another important property of the specification presented is that if the flow of messages in \( S \) can be represented by an acyclic graph, then \( \text{NMRSYST}(S) \) is an adequate generalized path. Such a result may be viewed as signifying that the specification of replication of modules we have presented does not lead to additional deadlock situations. We first state formally what an acyclic graph is.

An \( \text{NMR-} \)system \( S = (G, \phi) \) is said to be \emph{acyclic} iff the directed graph \( T_{S} \), obtained from \( G \) by adding arcs \( (p,q) \) for all \( p \in \text{In}_{G} \) and \( q \notin \phi(p) \) and by deleting circles representing nodes, is acyclic, where by a cycle we mean any sequence of vertices \( p_{1}, \ldots, p_{s} \) such that \( T_{S} \) contains the following arcs: \( (p_{1}, p_{2}), (p_{2}, p_{3}), \ldots, (p_{s-1}, p_{s}), (p_{s}, p_{1}) \).

For example, the \( \text{NMR-} \)system \( S \) specified in the Example is acyclic, and the corresponding graph \( T_{S} \) is shown in Fig. 5.

We are now in a position to prove the following theorem.
THEOREM 2. If $S = (G, \phi)$ is an acyclic NMR-system, then $P = NMR\text{SYST}(S)$ is adequate.

PROOF (sketch)

One can prove the theorem in two steps. First we show that:

\[(4.4) \quad \forall \tilde{x} \in VFS(P) \exists \tilde{y} \in VFS(P) \forall i, \tilde{x}iP_i \in Cyc(P_i)^* \]

i.e. $\tilde{x}$ is a history leading to the initial state of the system. The crucial points in proving (4.4) are:
(i) $S$ being acyclic, (ii) the presence of $(\text{reset}_{M,p}, \text{set}_{M,p})^*$ between $\text{set}_{M,p}$ and $\Omega_{n,p}$ in each single path $IN_{M,p}$ (this allows the flag $\text{set}_{M,p}$ to be released (if set) after each history $u$ satisfying $\Psi_{M,p}(u) = n-1$, and avoids an artificial deadlock situation).

In the second step we prove that starting from the 'initial state' it is possible to perform each event, i.e.

\[(4.5) \quad \forall a \in Ec(P) \exists v \in Vev(P)^*, va \in VFS(P). \]

Notice that (4.5) follows essentially from $S$ being acyclic, which implies that for every $p \in (In_G \cup Out_G)$ there is an external input port $q$ such that there is a directed path from $q$ to $p$ in $T_S$. Finally, from (4.4) and (4.5) we obtain the adequacy of $P$. \[\Box\]

5. Implementing replicated computations

The given specification of a distributed synchronization mechanism is not founded on, and does not suggest, any specific implementation. However, an efficient implementation should try to be reasonably fair and should ensure that communications are not delayed unreasonably often. In particular, it appears to be crucial to ensure: (1) a fair behaviour of the replicated processes; (2) the required response time in the case of a real time control system.

In this section two possible fair implementations will be briefly presented and discussed.

The system architecture considered here consists of a set of nodes, each one composed out of a processor, private memory, and input-output devices. These nodes are connected by a communication subsystem, which may include shared memory. The kernel code of the operating
system is replicated in the private memory of each processor, and each copy contains the local data structures for the dispatching of processes.

The first of the two proposed solutions is based on the notion of atomic actions [BeRa81]. The second solution utilizes a distributed algorithm for achieving cooperation between the instances of the kernel of the system nodes. Neither of the proposed solutions involves any planned scheduling of the task replications.

In implementing communication, it is convenient to specify a time interval indicating the maximum delay $\Delta$ between the sending and receiving of all the copies of a message from an NMR node. In general, $\Delta$ is equal to the scheduling delay plus the communications delay.

To illustrate clearly the effectiveness of the solutions, it will be assumed that faults can only corrupt messages, but that they do not cause messages to be lost, i.e. all the copies of a message reach the receivers within a finite time. It is worth noting that this assumption does not cause loss of generality - when the time interval $\Delta$ expires, any non-received message can be regarded as the reception of a message with an empty value.

5.1. Solution based on atomic actions

An action is a unit of work. It appears to be primitive to its surrounding environment. That is, an action appears to be atomic to other actions. Once begun, an action either completes by committing or fails by aborting. If an action aborts, it has no permanent effect on its environment. This must be supported by synchronization and recovery mechanisms that ensure that changes made by a failed action are undone and that partial results of failed actions are not incorporated in the results of other actions that eventually commit.

This solution requires the following assumption to be satisfied:

A.1. The message send operation is an atomic broadcast [CASD85]. This means that the communications involving the replications of the same port either completes by success or fails.

No assumption is made on communications related to distinct ports. The steps of this algorithm will be discussed taking into reference Fig.1. In particular, the sending of a message to the NMR node $N_3$ from the modules of $N_1$ proceeds as follows.

1. Each sending module records the message in the insertion buffers allocated on the nodes where the modules $M_1, \ldots, M_n$ of $N_3$ run. These communications are atomic.

2. After $n$ messages have been received from the same port, the kernel related to each $M_i$ of $N_3$ performs the decision algorithm. Before the single decision result is inserted in the appropriate buffer to be processed by $M_i$, a mark is added. This mark (for instance a sequence number local to the module) specifies the incoming ordering.
3. The non-deterministic strategy followed by the receiver modules consists of selecting the message with the smallest mark value.

If an NMR node has got several input ports an additional mechanism is needed. Suppose, for example, that input ports c and d of node \( N_2 \) in Fig. 1 receive \( n \) messages from the modules of node \( N_1 \) and \( N_2 \) respectively. In this case, in order to satisfy \( REQ \), each kernel \( K_1, \ldots, K_n \) of the nodes where the modules \( M_1, \ldots, M_n \) of \( N_2 \) run has to perform the decision algorithm according to a priority list referring to the senders. The priority of the senders is kept locally at each \( K_i \). After the message from a sender, say \( N_1 \), has been accepted, the lowest value is assigned to the priority of \( N_1 \). The aim of the priority list is to guarantee the fairness of the implementation.

The above solution satisfies the given specification. The modules of an NMR node service the input requests in an identical order. It is worth noting that there are no reliability problems in marking messages because marking is performed by the kernel of the node where the receiver process runs.

5.2. Using Decentralized Agreement

The preceding solution uses marks which imply a total order on the messages received from the various modules of an NMR node. In order to assure the same order for all the modules, we adopted atomic communications.

An alternative approach is possible: the kernel instances of each NMR node can agree (by a decentralized protocol) on the value of the mark which must be assigned to each received message. The kernel cooperation algorithm must be reliable, guaranteeing the agreement even in the presence of faulty nodes.

A solution using an algorithm with the necessary reliability requirement, namely the signed message algorithm of interactive consistency [LSP82], is discussed in [Man86]. It can be shown that this solution satisfies the given specification. In this case the order among the messages received by the modules of an NMR node is guaranteed by an agreement among the kernels \( K_1, \ldots, K_n \) on the set of messages which are to be ordered and the fact that every kernel of \( K_1, \ldots, K_n \) uses the same criterion for message ordering.

6. Concluding Remarks

A formal specification and correctness proof of a fully distributed protocol allowing the establishment of synchronization points among replicated computations has been discussed for a concurrent programming model consisting of communicating processes. In particular, this synchronization mechanism has been shown to be effective in preventing the occurrence of sequencing failures in NMR distributed systems. We have also given a formal specification of the synchronization requirements which must be satisfied in implementing replicated computations in nondeterministic models.
This work allows the replicated modular processes to be nondeterministic, and also permits the adoption of software fault tolerance techniques. Indeed, instead of merely running identical copies of a program, this approach gives the opportunity to run different implementations which satisfy the same specification. As shown in [AvKe84] this guards against software faults. Additional advantages of the proposed solution are: (1) it can provide fault tolerance support for a wide class of program; (2) it allows a considerable level of asynchrony between executions of program replications.

It is by no means trivial, nor is it within the scope of the paper, to evaluate our solution or to compare it to others in terms of performance; this may be the object of further work, together with the development of alternative solutions. Here we shall only note that the cost of our solution may be minimized by employing a special hardware unit dedicated to performing the synchronization phase. It is worth noting that since tools have been implemented which allow the efficient compilation into VLSI circuits of path expression specification [ACFM86], the given COSY specification turns out to be also suitable to provide our synchronization mechanism with hardware supports.

Acknowledgements

The authors are grateful to Brian Randell, Giuseppe Pappalardo, and the anonymous referees for their comments and suggestions. This work was supported by the Royal Signals and Radar Establishment of the U.K. Ministry of Defence, and the Science and Engineering Research Council.
References


