A CHARACTERIZATION OF ASYNCHRONOUS MESSAGE-PASSING

S.R.GOREGAOKAR*
Department of Computer Science & Engineering
Indian Institute of Technology, Powai
Bombay 400 076, India.

S.ARUN-KUMAR
Computer Science Group
Tata Institute of Fundamental Research
Homi Bhabha Road, Colaba
Bombay 400 005, India.

INTRODUCTION

ere have been several works in recent years which have attempted to opound a general theory of concurrency (e.g. /H 85/ ,/M 80/,/M 83/). st of them attempt to place the theory in communication. In most cases mmunication is represented by the symbols of an alphabet and the detail the actual communication (e.g. source, target, value etc.) are stracted away.

In this paper we do not attempt such a general theory. However, we dc lieve that a proper treatment of asynchronous message-passing is cessary since it underlies the implementation of all distributed systen nsider a network. It is usually organized as a set of layers /T 81/. each layer the communication assumptions (e.g. reliability, ordering c ssages) of the layer above are implemented. These implementations depe the assumptions implemented in the layers below. As a result at each yer it is necessary to be able to verify programs in terms of the sumptions below it.

It is then useful to have a uniform framework for verifying programs all layers of the network. Such a verification method applied at one

---

resent Address: Patni Computer Systems Pvt. Ltd., Unit No. 55, SDF 2,

layer will differ from that of another layer merely in the communication assumptions that it uses. In general, the assumptions of one layer are merely those of the layer immediately below with some "add-ons".

For example, in analysing the behaviours of communication protocols it is necessary to be able to define and identify high-level properties of the logical communication medium on which they are implemented. These properties should be "natural" and at the same time mathematically tractable so that the protocol may be checked against these properties and verified rigorously. Typically these properties may have to do with the temporal ordering of messages sent between processes, or certain high-level relations between the histories of communication of the various processes in a system.

The work reported here may best be understood in the context of a CSP-like language supporting asynchronous message-passing (instead of handshaking) and a compositional partial-correctness proof system along the lines of /S 83/. We regard the history of communications of a process as a suitable abstraction of its dynamic behaviour. We will be concerned only with finite behaviours of processes expressed in the CSP-like language. The results reported here may be neither surprising nor unknown to most workers in the area. Nevertheless we have not come across these results in the literature and the proofs are not very easy.

1.1 The Communication Model

We first outline our basic communication model. It will be clear later that one part of our characterization is, in fact, independent of it. Assume a network of n processes in which, each process may communicate with every other process. The communication medium is asynchronous, point-to-point and reliable. By "reliable" we mean that no messages are lost in the medium, nor are any spurious ones generated in it. Therefore when a

process P_i sends a message to another process P_j (the dotted arrows in Fig. 1), the message is sent along the communication layer (the continuous arrows) to the site of P_i. The message is guaranteed to reach and is "received" at the site of P_j. It is "accepted" by process P_j only if it is expecting a message from P_i. Otherwise the message is simply queued at site P_j. All messages from P_i to P_j are received and accepted by P_j in the order in which they are sent. This mechanism is realizable at some layer of the network /T 81/. No other assumptions are made about the global order of messages received from different processes.

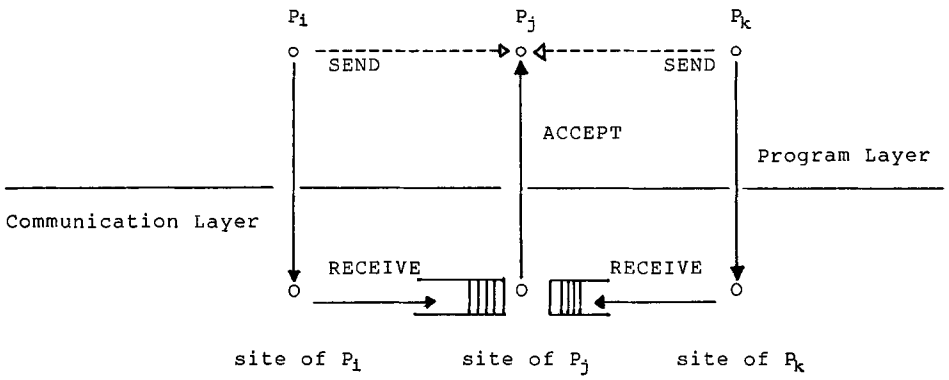

Fig. 1

1.2 Notations and Definitions

Let $N_n = \{1,2,\ldots,n\}$ and Z be the set of integers. A message is a triple $\langle i,j,v \rangle$ belonging to the set $N_n \times N_n \times Z$, where i is the index of the source process, j that of the target process ($i \neq j$) and v the value sent from P_i to P_j (or the value accepted by P_i from P_j). For i in N_n a history sequence (or simply history) h_i is a (finite) word on the alphabet M_i ($h_i \in M_i^*$) where

$$M_i = I_i \cup O_i$$
$$I_i = (N_n - \{i\}) \times \{i\} \times Z$$

$$O_i = \{i\} \times (N_n - \{i\}) \times Z$$

I_i is the set of input messages for P_i and O_i is the set of output messages. The <u>history vector</u> $\underset{\sim}{h}$ of a program is defined by

$$\underset{\sim}{h} = \langle h_i \mid i \in N_n \rangle$$

For every message $\mu = \langle i,j,v \rangle$ we define the functions

$$S(\mu) = i, \quad R(\mu) = j, \quad V(\mu) = v, \quad D(\mu) = \langle i,j \rangle$$

Some of the notations and functions used are defined below:

1. "[]" denotes the empty history and "." denotes catenation.
2. hd (h_i) gives the first message of a nonempty history,
   tl (h_i) gives the rest of the history h_i , i.e.

   $$h_i = hd(h_i).tl(h_i) \quad \text{for } h_i \neq []$$

   $|h_i|$ denotes the length of the sequence h_i,
   $\| A \|$ denotes the cardinality of a set A.
3. h_i/j := IF $h_i = []$ THEN []
   ELSEIF $S(hd(h_i)) = j \vee R(hd(h_i)) = j$ THEN
   $hd(h_i).(tl(h_i)/j)$
   ELSE $tl(h_i)/j$

   h_i/j is the subsequence of h_i representing only communications with P_j .
4. $h_i/\langle k,l \rangle$:= IF $h_i = []$ THEN []
   ELSEIF $D(hd(h_i)) = \langle k,l \rangle$ THEN $hd(h_i).(tl(h_i)/\langle k,l \rangle)$
   ELSE $tl(h_i)/\langle k,l \rangle$

where $i=k\neq l$ or $i=l\neq k$ and i,k,l in N_n . $h_i/\langle k,l \rangle$ is the subsequence of h_i containing messages of the form $\langle k,l,v \rangle$ for $v \in Z$.

5. I (h_i) := IF $h_i = []$ THEN []

ELSEIF $R(hd(h_i)) = i$ THEN $hd(h_i).I(tl(h_i))$

ELSE $I(tl(h_i))$

$I(h_i)$ is the sequence of messages accepted by P_i.

6. $O(h_i)$:= IF $h_i = []$ THEN $[]$

ELSEIF $S(hd(h_i)) = i$ THEN $hd(h_i).O(tl(h_i))$

ELSE $O(tl(h_i))$

$O(h_i)$ is the sequence of messages sent by P_i.

7. h'_i := IF $h_i = []$ $\vee$ $R(hd(h_i)) = i$ THEN $[]$

ELSE $hd(h_i).(tl(h_i)')$

h'_i is the sequence of messages sent by P_i before accepting any.

8. h''_i := IF $h_i = []$ $\vee$ $R(hd(h_i)) = i$ THEN h_i

ELSE $(tl(h_i))''$

h''_i is the sequence obtained by deleting h'_i from h_i. It is clear that every h_i may be expressed as

$$h_i = h'_i . h''_i$$

where $h'_i \in O_i^*$ and $h''_i \in \mu.M_i^* \cup \{[]\}$ with $\mu \in I_i$.

2. THE PREDICATE Compat

In a Soundararajan-Dahl-style proof-system (/SD 82/, /S 83/, /S 84a/, /S 84b/) the rule for parallel-composition of processes P_i (i in N_n) is given as follows.

$$\frac{\{p_i(\bar{X}_i, h_i) \wedge h_i = []\} \quad P_i\{q_i(\bar{X}_i, h_i)\}, \qquad i \in N_n}{\{\bigwedge_{i \in N_n} p_i(\bar{X}_i, h_i)\} \quad \| _{i \in N_n} P_i \{\bigwedge_{i \in N_n} q_i(\bar{X}_i, h_i) \wedge Compat(\underset{\sim}{h})\}}$$

where $\bar{X}_i$ denotes the local state of P_i. The predicate Compat ($\underset{\sim}{h}$) gives vital information regarding the properties of the communication medium. Put in another form, Compat restricts the possible histories that may be produced by the processes of a program because of the properties of the communication mechanism. The constraints imposed by the structure of the process are reflected in the local predicates p_i and q_i.

To illustrate what we mean we consider the following program written in a CSP-like notation with the input and output commands changed in order to reflect their asynchronous nature. That is, when a process has to send a value to another it simply sends it and continues its execution. A process that has to input the value waits (if necessary) till the value arrives, accepts it and continues execution.

```
P0 :: b:=true;  x:=1;  do true      -> P1!!x; P2!!x;  x:=x+2
                       [] b; P1??y  -> b  := false
                       [] ¬b; P2??y -> b  := true
                       od
||P1 :: do  P0??m -> P0!!m  od
||P2 :: do  P0??n -> P0!!(n+1)  od
```

In the above program, the following assertions are operationally valid and quite obvious.

A1. P_0 receives and accepts all odd numbers from P_1 and even numbers from P_2 .

A2. y eventually takes the values 1,2,3,... in that order.

A3. The first communication actions of P_0 are outputs to P_1 and P_2.

But these facts are not evident from P_0 alone. In particular, consider A3. The body of P_0 permits the first communication action of P_0 to be that of accepting an input value from P_1 . However it clearly is not possible in

any execution of the program.

It is therefore necessary to be able to restrict the history vector of the program in order to be able to infer facts like A3. Further, it is necessary to be able to draw a correspondence between the output message of one process and its acceptance by another (to infer A1 and A2). The need for a predicate Compat which characterizes the set of history vectors that are "compatible" is now clear.

The Compat predicate must take into account two important constraints. Firstly, it must be abstract enough to reflect the global non-determinism that is an intrinsic characteristic of loosely coupled processes executing at arbitrary finite (but positive) speeds. Secondly, it must contain enough information for proving partial correctness properties of the program.

We define below a function, Match, which first establishes the required correspondence between output and input messages in the history set. The function Match depends on another function, Remove, that removes the corresponding pair of output and input messages from the history vector. These functions pave the way for a recursive definition of Compat.

In the definitions we use "$\perp$" to mean "undefined" and i,j are the indices of distinct processes. The end of a proof is marked by "$\dashv$".

$$\text{Remove}(h_i, j) := \text{IF } h_i^! \neq [] \text{ THEN IF } R(\text{hd}(h_i)) = j \text{ THEN } \text{tl}(h_i) \text{ ELSE } \text{hd}(h_i).\text{Remove}(\text{tl}(h_i), j) \text{ ELSE } \perp$$

$$\text{Match}(\underset{\sim}{h}) := \{\underset{\sim}{h} \backslash \langle i,j \rangle \mid \text{hd}(h_i^! / \langle i,j \rangle) = \text{hd}(h_j^{"}),\ h_i^! / \langle i,j \rangle \neq [] \neq h_j^{"}, i \neq j\}$$

where $\underset{\sim}{h} \backslash \langle i,j \rangle := \langle h_k \backslash \langle i,j \rangle \mid k \in N_n \rangle$

and $h_k \backslash \langle i,j \rangle := \begin{cases} \text{Remove } (h_k, j) & \text{if } k = i \\ h_k' \,.\, tl(h_k'') & \text{if } k = j \\ h_k & \text{otherwise} \end{cases}$

The condition $hd(h_i'/\langle i,j\rangle) = hd(h_j'')$ ensures that any arbitrary "matching" pair is not selected. The choice of i and j is restricted by the fact that the message should occur before any input message in h_i and should be the first input message occuring in h_j. Notice that the Match function gives a set of history vectors. When $\|Match(\underset{\sim}{h})\| > 1$ it means that two or more processes could have accepted messages in any order. The cardinality of Match directly reflects the degree of global non-determinism at a given point in the execution of the program. This becomes clearer in lemmas 2.1 and 2.2.

In general we write $\underset{\sim}{h}^{\cdot}$ or $\underset{\sim}{h}^{+}$ to denote typical elements belonging to Match $(\underset{\sim}{h})$. Sometimes we may also write $\underset{\sim}{h} \rightarrow \underset{\sim}{h}^{\cdot}$ to mean $\underset{\sim}{h}^{\cdot} \in Match(\underset{\sim}{h})$.

Lemma 2.1: Let $\underset{\sim}{h}^{\cdot}, \underset{\sim}{h}^{+} \in Match(\underset{\sim}{h})$ and $\underset{\sim}{h}^{\cdot} \neq \underset{\sim}{h}^{+}$. Then $Match(\underset{\sim}{h}^{\cdot}) \cap Match(\underset{\sim}{h}^{+}) \neq \varphi$.

Proof: We give a constructive proof of the existence of a common element. Let

$\underset{\sim}{h}^{\cdot} = \underset{\sim}{h} \backslash \langle i_1, j_1 \rangle$ for some $i_1, j_1 \in N_n$, $i_1 \neq j_1$

and $\underset{\sim}{h}^{+} = \underset{\sim}{h} \backslash \langle i_2, j_2 \rangle$ for some $i_2, j_2 \in N_n$, $i_2 \neq j_2 \neq j_1$.

We have $hd(h_{i_1}' /\langle i_1, j_1 \rangle) = hd(h_{j_1}'')$ and $hd(h_{i_2}' /\langle i_2, j_2 \rangle) = hd(h_{j_2}'')$.

Since $j_1 \neq j_2$ we get $hd((\underset{\sim}{h}^{\cdot}_{j_2})'') = hd((h_{j_2} \backslash \langle i_1, j_1 \rangle)'') = hd(h_{j_2}'')$ and

$hd(h_{i_2} /\langle i_2, j_2 \rangle) = hd(h_{i_2}' /\langle i_2, j_2 \rangle)$. It is then clear that

$\underset{\sim}{h}^{\cdot +} = \underset{\sim}{h}^{\cdot} \backslash \langle i_2, j_2 \rangle \in Match(\underset{\sim}{h}^{\cdot})$ and $\underset{\sim}{h}^{+\cdot} = \underset{\sim}{h}^{+} \backslash \langle i_1, j_1 \rangle \in Match(\underset{\sim}{h}^{+})$. From the definition of Match it follows that $\underset{\sim}{h}^{\cdot +} = \underset{\sim}{h}^{+ \cdot}$.

Hence $\underset{\sim}{h}^{\cdot +} = \underset{\sim}{h}^{+ \cdot} \in Match(\underset{\sim}{h}^{\cdot}) \cap Match(\underset{\sim}{h}^{+})$. ⊣

We are now ready to define the predicate Compat.

$$Compat(\underset{\sim}{h}) := \left[\bigwedge_{i \in N_n} (h_i'' = [\,]) \right] \vee \left[Match(\underset{\sim}{h}) \neq \varphi \quad \wedge \quad \forall \underset{\sim}{h}^{\cdot} \in Match(\underset{\sim}{h}): Compat(\underset{\sim}{h}^{\cdot}) \right]$$

Two points need to be emphasized in the above definition.

1. Attention is mostly focussed on the input messages, becuase they are crucial in determining the course of program execution (e.g. whether it deadlocks, since deadlocks are possible only at input commands) and directly influence the global state of the program. The output messages on the other hand, do not affect the program state and hence their orders are immaterial beyond whatever the definition of Match stipulates.

2. The function Match imposes an order on the messages sent and accepted while at the same time it reflects the possibility of communications occuring in parallel.

Lemma 2.2: Given a history vector $\underset{\sim}{h}^0$ satisfying Compat($\underset{\sim}{h}^0$). There exists a finite sequence $\underset{\sim}{h}^0 \rightarrow \underset{\sim}{h}^1 \rightarrow \ldots \rightarrow \underset{\sim}{h}^r$ such that Match($\underset{\sim}{h}^r$) $= \varphi$. Further r ($r > 0$) and $\underset{\sim}{h}^r$ so obtained are unique.

Outline of proof: The proof is by induction on the number of input messages in $\underset{\sim}{h}^0$ given by $\sum_{i \in N_n} |I(h_i^0)|$. The base step of the induction is trivially proved by taking $r = 0$ and $\underset{\sim}{h}^r = \underset{\sim}{h}^0$. For the inductive step, the function Match has to be invoked. If $\underset{\sim}{h}^{m+1} \in$ Match($\underset{\sim}{h}^m$), $m \geq 0$, notice that $\underset{\sim}{h}^{m+1}$ has one input message less than $\underset{\sim}{h}^m$. Since there are only a finite number of input messages in $\underset{\sim}{h}^0$, the sequence terminates with $r = \sum_{i \in N_n} |I(h_i^0)|$. The uniqueness of r is thus guaranteed. The uniqueness of $\underset{\sim}{h}^r$ is evident from the fact that it consists of only unmatched output messages whose orders are preserved in the histories (by definition of Remove). -|

We have offered intuitive arguments to justify the definition of Compat. However it may be shown /Go 85/ that with this definition a sound and complete partial correctness proof system for the language is obtained.

3. ACYCLICITY

We now investigate a property which ought to be satisfied by all

reliable message-passing systems.

We first define a predicate Cyclic as follows:

$$\text{Cyclic}(\underset{\sim}{h}) := \exists \underset{\sim}{h}_{\alpha} \subseteq \underset{\sim}{h}: \underset{\sim}{h}_{\alpha} = \{h_{\alpha_\ell} \mid 0 \leq \ell < m\} \wedge \bigwedge_{\ell=0}^{m-1} (D(hd(h''_{\alpha_i})) = \langle \alpha_{i\oplus 1}, \alpha_i \rangle \wedge h'_{\alpha_{i\oplus 1}}/\alpha_i = [])$$

where $\oplus$ and $\ominus$ are used to denote addition and subtraction modulo m respectively. If the predicate holds the sequence $\langle h_{\alpha_\ell} \mid 0 \leq \ell < m \rangle$ so obtained is termed a cycle. We now prove some elementary properties involving Match, Cyclic and Compat.

Lemma 3.1: $\text{Cyclic}(\underset{\sim}{h}) \Rightarrow \forall \underset{\sim}{h}' \in \text{Match}(\underset{\sim}{h}): \text{Cyclic}(\underset{\sim}{h}')$

Proof: Let $\langle h_{\alpha_\ell} \mid 0 \leq \ell < m \rangle$ be a cycle and $C = \{h_{\alpha_\ell} \mid 0 \leq \ell < m\}$. If $\text{Match}(\underset{\sim}{h}) = \varphi$ there is nothing to prove. Assume $\underset{\sim}{h}' \in \text{Match}(\underset{\sim}{h})$. Then $\underset{\sim}{h}' = \underset{\sim}{h} \backslash \langle i,j \rangle$ for some $i,j \in N_n$ and $i \neq j$. We have

$h'_i = \text{Remove}(h_i, j)$

$h'_j = h'_j.tl(h''_j)$

$h'_k = h_k$ for $i \neq k \neq j$

Clearly $h_j \notin C$ (otherwise $\underset{\sim}{h}' \notin \text{Match}(\underset{\sim}{h})$). If $h_i \notin C$ then $\langle h'_{\alpha_\ell} \mid 0 \leq \ell < m \rangle$ $(= \langle h_{\alpha_\ell} \mid 0 \leq \ell < m \rangle)$ is a cycle. If $h_i \in C$ let $i = \alpha_\ell$ for some ℓ, $0 \leq \ell < m$. Hence $h'_{\alpha_\ell}/\alpha_{\ell\ominus 1} = []$ and $h'_{\alpha_\ell} = \text{Remove}(h_{\alpha_\ell}, j)$ which implies that $(h'_{\alpha_\ell})'/\alpha_{\ell\ominus 1} = []$. Therefore $\langle h'_{\alpha_\ell} \mid 0 \leq \ell < m \rangle$ is still a cycle. ⊣

Notice that the converse of this lemma is not true. Consider the history vector

$h_1 = \langle 1,2,x \rangle . \langle 2,1,y \rangle . \langle 1,2,z \rangle$

$h_2 = \langle 1,2,x \rangle . \langle 1,2,z \rangle . \langle 2,1,y \rangle$

with $h'_1 = \langle 2,1,y \rangle . \langle 1,2,z \rangle$

$h'_2 = \langle 1,2,z \rangle . \langle 2,1,y \rangle$

While $\underset{\sim}{h}' \in \text{Match}(\underset{\sim}{h})$ and $\text{Cyclic}(\underset{\sim}{h}')$ holds, $\text{Cyclic}(\underset{\sim}{h})$ does not. We now relate Compat and Cyclic by the following lemma.

Lemma 3.2: $\text{Compat}(\underset{\sim}{h}) \Rightarrow \neg\text{Cyclic}(\underset{\sim}{h})$.

Proof: By induction on $r = \sum_{i \in N_n} |I(h_i)|$. For $r = 0$ the lemma holds trivially. If $r \neq 0$ we have by definition of Compat

$$\text{Compat}(\underset{\sim}{h}) \wedge r>0 \Rightarrow \text{Match}(\underset{\sim}{h}) \neq \varphi \wedge \forall \underset{\sim}{h}' \in \text{Match}(\underset{\sim}{h}): \text{Compat}(\underset{\sim}{h}')$$

Hence by the induction hypothesis and lemma 3.1 $\neg\text{Cyclic}(\underset{\sim}{h}')$ holds. ⊣

The previous two lemmas and the example suggest that a stronger predicate needs to be defined for which the analogue of lemma 3.1 and its converse will both hold. Define

$$\omega\text{-acyclic}(\underset{\sim}{h}) := \neg\text{Cyclic}(\underset{\sim}{h}) \wedge \forall \underset{\sim}{h}' \in \text{Match}(h): \omega\text{-acyclic}(\underset{\sim}{h}')$$

From the example it is clear that the absence of a cycle needs to be checked at every level in $\underset{\sim}{h}$ in order for it to be a compatible history vector. The above definition is an attempt to capture this notion of acyclicity. As may be seen from the following example ω-acyclicity is not merely a negation of Cyclic.

$h_1 = \langle 1,2,w\rangle . \langle 2,1,x\rangle . \langle 1,3,z\rangle$

$h_2 = \langle 1,2,w\rangle . \langle 3,2,y\rangle . \langle 2,1,x\rangle$

$h_3 = \langle 1,3,z\rangle . \langle 3,2,y\rangle$

Here $\text{Cyclic}(\underset{\sim}{h})$ and ω-$\text{acyclic}(\underset{\sim}{h})$ are both false. However, ω-$\text{acyclic}(\underset{\sim}{h})$ is false because $\text{Cyclic}(\underset{\sim}{h}')$ is true, where $\underset{\sim}{h}'$ is given by

$h'_1 = \langle 2,1,x\rangle . \langle 1,3,z\rangle$

$h'_2 = \langle 3,2,y\rangle . \langle 2,1,x\rangle$

$h'_3 = \langle 1,3,z\rangle . \langle 3,2,y\rangle$

Proposition 3.3: Let $\text{Match}(\underset{\sim}{h}) \neq \varphi$. Then ω-$\text{acyclic}(\underset{\sim}{h})$ iff there exists $\underset{\sim}{h}' \in \text{Match}(\underset{\sim}{h})$ such that ω-$\text{acyclic}(\underset{\sim}{h}')$.

Proof: (=>) Follows from the definition of ω-acyclic.

(<=) Obviously $\neg\text{Cyclic}(\underset{\sim}{h})$ holds. The result follows from the claim:

$$\exists \underset{\sim}{h}' \in \text{Match}(\underset{\sim}{h}): \omega\text{-acyclic}(\underset{\sim}{h}') \Rightarrow \forall \underset{\sim}{h}^+ \in \text{Match}(\underset{\sim}{h}): \omega\text{-acyclic}(h^+)$$

Proof of Claim: Since $\text{Match}(\underset{\sim}{h}) \neq \varphi$ we prove it by induction on r, $r \geq 1$. For

r=1 the result is trivial. For $r > 1$ and $\|Match(\underset{\sim}{h})\| = 1$ also it is trivial. So let $\underset{\sim}{h}^{+} \in Match(\underset{\sim}{h})$ be distinct from $\underset{\sim}{h}^{\cdot}$. Hence $Match(\underset{\sim}{h}^{\cdot})$ and $Match(\underset{\sim}{h}^{+})$ have a common element $\underset{\sim}{h}^{\cdot+}$. From ω-acyclic$(\underset{\sim}{h}^{\cdot})$ it follows that ω-acyclic$(\underset{\sim}{h}^{\cdot+})$ holds. Hence we have

$$\exists \underset{\sim}{h}^{\cdot+} \in Match(\underset{\sim}{h}^{+}): \ \omega\text{-acyclic}(\underset{\sim}{h}^{\cdot+}) \qquad \ldots (1)$$

Obviously $\neg$Cyclic$(\underset{\sim}{h}^{+})$ holds $\qquad \ldots (2)$

From (1) and the induction hypothesis we have

$$\forall \underset{\sim}{h}^{++} \in Match(\underset{\sim}{h}^{+}): \ \omega\text{-acyclic}(\underset{\sim}{h}^{++}) \qquad \ldots (3)$$

From (2) and (3) we get ω-acyclic$(\underset{\sim}{h}^{+})$. ⊣

Lemma 3.4: Compat$(\underset{\sim}{h}) \Rightarrow \omega$-acyclic$(\underset{\sim}{h})$.

Proof: By induction on r. When r=0 there is nothing to prove. Assume $r > 0$. Therefore $Match(\underset{\sim}{h}) \neq \varphi$ and for every $\underset{\sim}{h}^{\cdot} \in Match(\underset{\sim}{h})$ we have Compat$(\underset{\sim}{h}^{\cdot})$. By the induction hypothesis we have ω-acyclic$(\underset{\sim}{h}^{\cdot})$. Further by lemma 3.2 we have $\neg$Cyclic$(\underset{\sim}{h})$. Hence by definition ω-acyclic$(\underset{\sim}{h})$ holds. ⊣

Note however that in our attempt to characterise Compat we have not fully succeeded since the converse of lemma 3.4 does not hold. This is evident from the following example.

$$h_1 = \langle 1,2,x \rangle . \langle 1,2,y \rangle$$

$$h_2 = \langle 1,2,y \rangle . \langle 1,2,x \rangle$$

with $x \neq y$. It is clear from this example that ω-acyclicity is only a necessary condition, not a sufficient one. It may also have been noticed that in our definitions of Cyclic and ω-acyclic, we have used only the fact that there is a time delay between the sending of a message and its acceptance by the target process. In fact ω-acyclicity may be used to prove the assertion A3 in Section 2. The features of our communication model (e.g. reliability, order-preservation) have not been used. The next section uses these.

4. PREFIX PROPERTY AND THE CHARACTERIZATION

The counter-example given in the last section suggests that we need to use the two features of our communication model -- reliability and order-preservation -- in order to complete the characterization. The two features may be combined into a single one viz. the prefix property. Define

$$h_i \leq h_j := h_i = [] \quad \vee \quad [\, hd(h_i) = hd(h_j) \wedge tl(h_i) \leq tl(h_j)]$$

If $h_i \leq h_j$ we say "h_i is an initial sub-sequence of h_j". Define the prefix property as

$$\text{Prefix}(\underset{\sim}{h}) := \bigwedge_{i,j \in N_n} [\, h_j / \langle i,j \rangle \leq h_i / \langle i,j \rangle]$$

We then have the following lemma.

Lemma 4.1: For $\underset{\sim}{h}' \in \text{Match}(\underset{\sim}{h})$, $\text{Prefix}(\underset{\sim}{h})$ iff $\text{Prefix}(\underset{\sim}{h}')$.

Proof: $\underset{\sim}{h}' = \underset{\sim}{h} \setminus \langle i,j \rangle$ for some distinct $i,j \in N_n$. Therefore

$h'_i = \text{Remove}(h_i, j)$

$h'_j = h_j.tl(h''_j)$

$h'_k = h_k$ for $i \neq k \neq j$

It is clear that $h'_k / \langle \ell,m \rangle = h_k / \langle \ell,m \rangle$ for all (k,ℓ,m) in $N_n \times N_n \times N_n$, such that $\ell \neq m$ and either $k = \ell$ or $k = m$ (but not both) except when

$$(k,\ell,m) = (i,i,j) \text{ or } (k,\ell,m) = (j,i,j)$$

We have only to show that

$$h'_j / \langle i,j \rangle \leq h'_i / \langle i,j \rangle \quad \text{iff} \quad h_j / \langle i,j \rangle \leq h_i / \langle i,j \rangle \qquad \ldots (1)$$

Clearly $h'_i / \langle i,j \rangle = (\text{Remove}(h_i,j)) / \langle i,j \rangle$

$$= \text{Remove}(h_i / \langle i,j \rangle, j)$$

$$= tl(h_i / \langle i,j \rangle) \qquad \ldots (2)$$

and $h'_j / \langle i,j \rangle = [\, h_j.tl(h''_j)\,] / \langle i,j \rangle$

$$= [\, h_j / \langle i,j \rangle] \,.\, [(tl(h''_j)) / \langle i,j \rangle]$$

$$= tl(h''_j) / \langle i,j \rangle \quad \text{since } h'_j / \langle i,j \rangle = []$$

$$= tl(h''_j / \langle i,j \rangle) \quad \text{since } D(hd(h''_j)) = \langle i,j \rangle \qquad \ldots (3)$$

(1) follows from (2), (3) and the definition of "$\leq$". ⊣

Proposition 4.2: Compat ($\underset{\sim}{h}$) $\Rightarrow$ Prefix($\underset{\sim}{h}$).

Proof: By induction on r. When r=0 there is nothing to prove (since for every i,j, $h_j / \langle i,j \rangle = [] \leq h_i / \langle i,j \rangle$). If r >0 we have for each $\underset{\sim}{h}' \in$ Match($\underset{\sim}{h}$), Compat($\underset{\sim}{h}'$) holds. By the induction hypothesis and lemma 4.1 we have Prefix ($\underset{\sim}{h}$). ⊣

Here again the reader may notice that the prefix property is not sufficient. For example consider

$$h_1 = \langle 2,1,x \rangle . \langle 1,2,y \rangle$$
$$h_2 = \langle 1,2,y \rangle . \langle 2,1,x \rangle$$

where Prefix($\langle h_1, h_2 \rangle$) holds but Compat($\langle h_1, h_2 \rangle$) does not. But this example suggests that perhaps acyclicity and the prefix property may together characterise Compat. It is indeed the case as the following proposition clearly shows.

Proposition 4.3: ω-acyclic($\underset{\sim}{h}$) $\wedge$ Prefix($\underset{\sim}{h}$) $\Rightarrow$ Compat($\underset{\sim}{h}$).

Proof: By induction on r. When r=0 the result follows trivially. If r >0 then by Prefix($\underset{\sim}{h}$) we have Match($\underset{\sim}{h}$)$\neq$[] and for every $\underset{\sim}{h}' \in$ Match($\underset{\sim}{h}$), ω-acyclic($\underset{\sim}{h}'$) $\wedge$ Prefix($\underset{\sim}{h}'$) holds. By induction hypothesis the result follows. ⊣

Theorem 4.4: Compat($\underset{\sim}{h}$) $\Leftrightarrow$ ω-acyclic($\underset{\sim}{h}$) $\wedge$ Prefix($\underset{\sim}{h}$).

Proof: Follows from lemma 3.4 and propositions 4.2 and 4.3. ⊣

Theorem 4.5:

$$\text{Compat}(\underset{\sim}{h}) := \Big[\bigwedge_{i \in N_n} h''_i = [] \Big] \vee \Big[\text{Match}(\underset{\sim}{h}) \neq \varphi \wedge \exists \underset{\sim}{h}' \in \text{Match}(\underset{\sim}{h}) : \text{Compat}(\underset{\sim}{h}') \Big]$$

is an equivalent definition of Compat.

Proof: Follows from proposition 3.3, lemma 4.1 and theorem 4.4. ⊣

Theorem 4.5 asserts that it is enough to perform the check on only one element in Match($\underset{\sim}{h}$).

5. AN EXAMPLE : THE WELFARE CROOK

We present a distributed solution to a generalization of the "welfare crook" problem /Gr 81/ . The problem is as follows: Given n unbounded lists of numbers (sorted in ascending order), to find a number that occurs in all the lists. Associated with each list L_i is a process P_i that searches through L_i. In each P_i , $i \in N_n$, "*!!$L_i[k_i]$" is an abbreviation for the sequence of statements

"P_{i+1} !!$L_i[k_i]$; ...; P_n!!$L_i[k_i]$; P_1!!$L_i[k_i]$; ...; P_{i-1}!!$L_i[k_i]$".

We have to prove that (if and) when the program terminates, for every i,j in N_n, $L_i[k_i] = L_j[k_j]$. The program to do this task is given below.

$\|_{i \in N_n} P_i$ :: $S_i := \varphi$; $k_i := 1$; *!!$L_i[k_i]$;

do

$\square_{\substack{j \in N_n \\ j \neq i}}$ $S_i \neq N_n - \{i\}$; P_j??y ->

if y < $L_i[k_i]$ -> $S_i := S_i - \{j\}$

[] y = $L_i[k_i]$ -> $S_i := S_i + \{j\}$

[] y > $L_i[k_i]$ ->

$S_i := \varphi$; do y > $L_i[k_i]$ -> $k_i := k_i + 1$ od;

if y = $L_i[k_i]$ -> $S_i := \{j\}$

[] y < $L_i[k_i]$ -> skip

fi; *!!$L_i[k_i]$

fi

od.

We use $\mathfrak{M}(h_i)$ to denote the last message of h_i and $B_i(k_i)$ to denote the sequence

$\langle i,i+1,L_i[k_i] \rangle . \langle i,i+2,L_i[k_i] \rangle . \ldots . \langle i,i-1,L_i[k_i] \rangle$.

From the individual processes it is clear (see /S 83/ for the relevant proof rules) that for every $i \in N_n$, $0(h_i)$ consists of a catenation of sequences of the form $B_i(k_i)$. Let β_i denote $0(h_i)$ and Body(β_i) denote

the sequence obtained from β_i by deleting the last n-1 messages(viz. $B_i(k_i)$ for some k_i). Now define the predicates

$$Q_i(\beta_i, k_i) \equiv |\beta_i| \geq n-1 \wedge \left[|\beta_i| = n-1 \Rightarrow (k_i = 1 \wedge \beta_i = B_i(k_i))\right] \wedge \left[|\beta_i| > n-1 \Rightarrow \exists k_i' : 1 \leq k_i' < k_i : Q_i(\text{Body}(\beta_i), k_i')\right]$$

$$R_i(h_i, k_i) \equiv \bigwedge_{j \in S_i} (V(\Omega(h_i/\langle j,i\rangle)) = L_i[k_i])$$

$$\text{Ord}_i(L_i) \equiv \forall k_i, k_i' \in N : k_i' < k_i \Leftrightarrow L_i[k_i'] < L_i[k_i']$$

It may be checked that $Q_i \wedge R_i \wedge \text{Ord}_i$ is a loop invariant for each P_i. The postcondition of the program is then given by

$$\left(\bigwedge_{i \in N_n} [S_i = N_n - \{i\} \wedge Q_i(\beta_i, k_i) \wedge R_i(h_i, k_i) \wedge \text{Ord}_i(L_i)]\right) \wedge \text{Compat}(\underset{\sim}{h}).$$

By proposition 4.2 if $\mu = \Omega(h_j/\langle j,i\rangle)$ and $\mu' = \Omega(h_i/\langle j,i\rangle)$ for $i \neq j$ then $V(\mu) = L_j[k_j]$ and $V(\mu') = L_j[k_j']$ for some k_j'. By $Q_j(\beta_j, k_j)$ and the fact that μ and μ' may not be distinct we get $V(\mu') \leq V(\mu)$ and hence $L_j[k_j'] \leq L_j[k_j]$. By $R_i(h_i, k_i)$ and $S_i = N_n - \{i\}$ we get $V(\mu') = L_i[k_i]$. Hence $L_i[k_i] \leq L_j[k_j]$ for all $i,j \in N_n$ and $i \neq j$, from which we conclude that $L_i[k_i] = L_j[k_j]$ for all $i,j \in N_n$.

6. IN RETROSPECT

Our characterization has expressed the compatibility checking predicate in terms of two simple properties -- acyclicity and prefix property. A closer look at these properties shows that acyclicity is a general property of all communication mechanisms i.e. on any layer of a network acyclicity would have to be satisfied. Loosely speaking, in the case of two processes (n=2) acyclicity merely states that "a process cannot possibly receive a reply to its message before it has sent the message". ω-acyclic is a generalization of this property to an arbitrary number of processes and to arbitrary depths within the histories of the processes.

The second property viz. the prefix property is much more dependent on

the communication model and is thus directly related to the assumptions of reliability and order-preservation.

In fact, for any communication mechanism that does not produce ghost messages it should be possible to express Compat in terms of ω-acyclic and another predicate that reflects the communication model. For example, consider the Compat predicate for CSP /S 83/ . It may be written (in our notation) as

$$\text{Compat}(\underset{\sim}{h}) \;:=\; \bigwedge_{i \in N_n} [\, h_i = [\,] \,] \;\wedge$$

$$[\, \exists h_i, h_j \in \underset{\sim}{h} :\; i \neq j \;\wedge\; \text{hd}(h_i) = \text{hd}(h_j) \;\wedge\; \text{Compat}(\underset{\sim}{h}^{\bullet}) \,]$$

where for $k \in N_n$ $h_k^{\bullet} := h_k$ if $i \neq k \neq j$ and $\text{tl}(h_k)$ otherwise.
It may be alternatively expressed as follows.

$$\text{Compat}(\underset{\sim}{h}) \;\Leftrightarrow\; \omega\text{-acyclic}(\underset{\sim}{h}) \;\wedge \bigwedge_{i,j \in N_n} [\, h_i/j = h_j/i \,]$$

where the prefix property has been replaced by a stronger predicate (which implies it) in order to reflect the synchronized handshaking mechanism. For a model that permits message losses but is order-preserving it is only necessary to weaken the prefix property to accommodate the "holes" in the histories.

As an aside we may also mention that in lemma 2.1 it is possible to prove that there is a unique $\underset{\sim}{h}^{\bullet +}$ that is common to $\text{Match}(\underset{\sim}{h}^{\bullet})$ and $\text{Match}(\underset{\sim}{h}^{+})$.

Acknowledgement: We are grateful to Mr. S.D. Sherlekar of IIT Bombay for having made this collaboration between the two authors possible. We are also grateful to the referees for having helped to improve the presentation of the paper. Margaret D'Souza did an excellent job of typing a difficult manuscript.

REFERENCES:

1. /Go 85/ S.R. Goregaokar: "Formal Semantics of Broadcasting Sequential Processes", M. Tech. Dissertation, IIT Bombay, 1985.
2. /Gr 81/ D. Gries: "The Science of Programming", Springer-Verlag, 1981.
3. /H 78/ C.A.R. Hoare: "Communicating Sequential Processes", CACM, Vol.21, No.8, 1978.
4. /H 85/ C.A.R. Hoare: "Communicating Sequential Processes", Prentice-Hall, 1985.
5. /M 80/ R. Milner: "A Calculus of Communicating Systems", Lecture Notes in Computer Science 92, Springer-Verlag, 1980.
6. /M 83/ R. Milner: "Calculi for Synchrony and Asynchrony", Theoretical Computer Science, Vol.25, No.3, July 1983.
7. /S 83/ N. Soundararajan: "Correctness proofs of CSP programs", Theoretical Computer Science, Vol.24, No.2, July 1983.
8. /S 84a/ N. Soundararajan: "A Proof Technique for Parallel Programs", Theoretical Computer Science, Vol.31, No.1-2, May 1984.
9. /S 84b/ N. Soundararajan: "Axiomatic Semantics of Communicating Sequential Processes", ACM-TOPLAS, Vol.6, No.4, October 1984.
10. /SD 82/ N. Soundararajan, O.J. Dahl: "Partial Correctness of Communicating Sequential Processes", Research Report, University of Oslo, 1982.
11. /T 81/ A.S. Tanenbaum: "Computer Networks", Prentice-Hall, 1981.