INFORMATION AND CONTROL 63, 200–216 (1984)

# On the Progress of Communication between Two Finite State Machines

M. G. GOUDA

*Department of Computer Sciences, University of Texas at Austin*

E. G. MANNING

*Department of Computer Science, University of Waterloo*

AND

Y. T. YU

*Department of Computer Sciences, University of Texas at Austin.*

We consider the following problem concerning any two finite state machines $M$ and $N$ that exchange messages via two 1-directional channels. "Is there a positive integer $K$ such that the communication between $M$ and $N$ over $K$-capacity channels is guaranteed to progress indefinitely?" The problem is shown to be undecidable in general. For a practical class of communicating machines, the problem is shown to be decidable, and the decidability algorithm is polynomial. We also discuss some sufficient conditions for the problem to have a positive answer; these sufficient conditions can be checked for the given $M$ and $N$ in polynomial time. We apply the results to some practical protocols to show that their communications will progress indefinitely. 

## I. INTRODUCTION

The model of communicating finite state machines is an abstraction of sequential processes which communicate exclusively by exchanging messages. The abstraction is achieved by suppressing the local data structures and internal operations of the processes, and representing each of them by its possible sequences of sending and receiving operations with other processes. This abstract model has been useful in the specification (Danthine, 1980; Sunshine, 1981), analysis (Bochmann, 1978; West 1978; Brand and Zafiropulo, 1983; Gouda and Yu, 1984a), and synthesis (Zafiropulo *et al.*, 1980; Gouda and Yu, 1984b) of communication protocols. But its major impact has been in characterizing some important



0019-9958/84 $3.00

communication progress properties such as boundedness, and freedom from deadlocks and unspecified receptions.

In this paper, we consider the general problem of communication progress between two finite state machines, and discuss its relationship to the above progress properties. We show that the problem is undecidable in general and present some special cases for which the problem is decidable by polynomial algorithms.

The paper is organized as follows. The model of communicating finite state machines is presented in Section II; then the communication progress problem is discussed and shown to be undecidable in Section III. In Section IV, the problem is shown to be decidable by a polynomial algorithm for a special class of communicating machines called alternating machines. In Sections V and VI, we discuss two sets of sufficient conditions to ensure that the problem has a positive answer. Concluding remarks are in Section VII.

## II. Networks of Communicating Machines

A *communicating machine* $M$ is a finite directed labeled graph with two types of edges, namely *sending* and *receiving edges*. A sending (receiving) edge is labeled send($g$) (receive($g$)) for some *message* $g$ in a finite set $G$ of messages. One of the nodes in $M$ is identified as its initial node, and each node is reachable by a directed path from the initial node. Each node in $M$ has at least one outgoing edge. A node in $M$ whose outgoing edges are all sending (receiving) edges is called a *sending* (*receiving*) *node*; otherwise it is called a *mixed node*. If the outgoing edges of each node in $M$ have distinct labels, then $M$ is called *deterministic*; otherwise it is called *nondeterministic*.

Let $M$ and $N$ be two communicating machines with the same set $G$ of messages; the pair $(M, N)$ is called a *network* of $M$ and $N$. A *state* of $(M, N)$ is a four-tuple $[v, w, x, y]$, where $v$ and $w$ are two nodes in $M$ and $N$ respectively, and $x$ and $y$ are two strings over the messages in $G$. Informally, a state $[v, w, x, y]$ means that the executions of $M$ and $N$ have reached nodes $v$ and $w$ respectively, while the input channels of $M$ and $N$ have the message sequences $x$ and $y$ respectively.

The *initial state* of network $(M, N)$ is $[v_0, w_0, E, E]$, where $v_0$ and $w_0$ are the initial nodes in $M$ and $N$ respectively, and $E$ is the empty string.

Let $s = [v, w, x, y]$ be a state of $(M, N)$; and let $e$ be an outgoing edge of node $v$ or $w$. A state $s'$ is said to *follow s over e* iff one of the following four conditions is satisfied:

(i) $e$ is a sending edge, labeled send($g$), from $v$ to $v'$ in $M$, and $s' = [v', w, x, y \cdot g]$, where "·" is the concatenation operator.

(ii) $e$ is a sending edge, labeled send($g$), from $w$ to $w'$ in $N$, and $s' = [v, w', x \cdot g, y]$.

(iii) $e$ is a receiving edge, labeled receive($g$), from $v$ to $v'$ in $M$, and $x = g \cdot x'$ and $s' = [v', w, x', y]$.

(iv) $e$ is a receiving edge, labeled receive($g$), from $w$ to $w'$ in $N$, and $y = g \cdot y'$ and $s' = [v, w', x, y']$.

Let $s$ and $s'$ be two states of network $(M, N)$, $s'$ *follows* $s$ iff there is a directed edge $e$ in $M$ or $N$ such that $s'$ follows $s$ over $e$.

Let $s$ and $s'$ be two states of $(M, N)$; $s'$ is *reachable from* $s$ iff $s = s'$ or there exists states $s_1, ..., s_r$ such that $s = s_1$, $s' = s_r$, and $s_{i+1}$ follows $s_i$ for $i = 1, ..., r-1$.

A state $s$ of network $(M, N)$ is said to be *reachable* iff it is reachable from the initial state of $(M, N)$.

The communication of a network $(M, N)$ is said to be *bounded by* a nonnegative integer $K$ iff every reachable state $[v, w, x, y]$ of $(M, N)$ is such that $|x| \leqslant K$ and $|y| \leqslant K$, where $|x|$ is the number of messages in string $x$. The communication of $(M, N)$ is *bounded* iff there exists a nonnegative integer $K$ such that the communication is bounded by $K$. If there is no such $K$, the communication is said to be *unbounded.*

A reachable state $[v, w, x, y]$ of $(M, N)$ is a *deadlock state* iff (i) both $v$ and $w$ are receiving nodes, and (ii) $x = y = E$ (the empty string). If no reachable state of $(M, N)$ is a deadlock state, then the communication of $(M, N)$ is said to be *deadlock-free.*

A reachable state $[v, w, x, y]$ of $(M, N)$ is an *unspecified reception state* iff one of the following two conditions is satisfied:

(i) $x = g_1 \cdot g_2 \cdot \cdots \cdot g_k$ $(k \geqslant 1)$; and $v$ is a receiving node and none of its outgoing edges is labeled receive $(g_1)$.

(ii) $y = g_1 \cdot g_2 \cdot \cdots \cdot g_k$ $(k \geqslant 1)$; and $w$ is a receiving node and none of its outgoing edges is labeled receive $(g_1)$.

If no reachable state of $(M, N)$ is an unspecified reception state, then the communication of $(M, N)$ is said to be *free from unspecified receptions.*

## III. The Communication Progress Problem

In this section, we state the communication progress problem for a network of two machines, and argue that it is undecidable in general.

Let $K$ be a nonnegative integer. The *K-reachable set* $R_K$ of a network $(M, N)$ is the set of all reachable states $[v, w, x, y]$ of $(M, N)$ such that $|x| \leqslant K$ and $|y| \leqslant K$. Informally, $R_K$ is the set of all reachable states of network $(M, N)$, where each of the two channels between $M$ and $N$ has a finite capacity of $K$. The next lemma follows immediately.

LEMMA 1. *Let $R$ be the set of all reachable states of a network $(M, N)$, and let $R_J$ and $R_K$ be the $J$- and $K$-reachable sets of $(M, N)$:*

(i) $R = \bigcup_{J=0}^{\infty} R_J$.

(ii) If $J \leqslant K$, *then $R_J$ is a subset of $R_K$.*

(iii) If $J < K$ *and $R_J = R_K$, then $R = R_J$.* ▮

Let $R_K$ be a $K$-reachable set of a network $(M, N)$. A state $s = [v, w, x, y]$ of $(M, N)$ is an *over flow state in $R_K$* iff one of the following two conditions is satisfied:

(i) Node $v$ has an outgoing sending edge, and $|y| = K$.

(ii) Node $w$ has an outgoing sending edge, and $|x| = K$.

A state $s$ of a network $(M, N)$ is a *nonprogress state in $R_K$* iff $s$ is a deadlock state, an unspecified reception state, or an overflow state in $R_K$. Otherwise, $s$ is called a *progress state in $R_K$*.

The last definition needs some justification. Consider the case of two machines $M$ and $N$ communicating over two $K$-capacity channels. If the network reaches a deadlock state, then indeed neither $M$ nor $N$ can progress any further. Thus, a deadlock state is a nonprogress state. If the network reaches an unspecified reception state, then one of the two machines is at a receiving node where the current "head" message in its input channel is not expected; and so it cannot progress any further. The other machine may still be able to progress; but since each progress step is accompanied by either receiving one message or sending one message, the other machine can progress at most $2K-1$ steps (where $K$ is the channel capacity), then it must stop forever. Thus, an unspecified reception state is a nonprogress state. If the network reaches an overflow state, then one of the two machines is at a node where it can send one message to its output channnel which is currently full. Obviously, this machine should not be "allowed" to progress any further. As before, the other machine can progress at most $2K$ steps after this point; then it must stop forever. Thus, an overflow state is a nonprogress state. Now, consider the argument in the other direction. Assume that $M$ and $N$, communicating over $K$-capacity channels, have reached a state $s$ after which no further progress is possible. In state $s$, each of the two machines must be in one of the following "local states":

(i) The machine is at a receiving node while its input channel is empty.

(ii) The machine is at a receiving node where the current head message in its input channel is not expected.

(iii) The machine is at a sending or mixed node while its output channel is full.

If both machines are in local states of type (i), then the nonprogress state $s$ is a deadlock state. If one machine is in a local state of type (ii), then the nonprogress state $s$ is an unspecified reception state. If one machine is in a local state of type (iii), then the nonprogress state $s$ is an overflow state. This completes our justification for the above definition of a nonprogress state.

In this paper, we address the following *communication progress problem*: "Given two communicating machines $M$ and $N$, is there a nonnegative integer $K$ such that each state in the $K$-reachable set of network $(M, N)$ is a progress state?" Notice that if an instance of this problem has a positive answer, then it is possible to determine the smallest $K$, denoted $K_{\min}$, for which this instance has a positive answer. This is because, in this case, the set $R$ of all reachable states of network $(M, N)$ is finite as can be shown from Theorem 1 (below). Therefore $K_{\min}$ can be computed as follows:

$$K_{\min} = \max_{[v,w,x,y] \text{in} R}(|x|, |y|).$$

The following theorem shows that the communication progress problem is equivalent to another problem concerning the communication of $M$ and $N$ over infinite capacity channels.

THEOREM 1. *Let $M$ and $N$ be two communicating machines. There exists a positive integer $K$ such that each state in the $K$-reachable set of network $(M, N)$ is a progress state iff the communication of $(M, N)$ is bounded, and free from deadlocks and unspecified receptions.*

*Proof.* (If) Assume that the communication of $(M, N)$ is bounded by some positive integer $K$. Therefore, the $K$-reachable set $R_K$ of network $(M, N)$ = the set $R$ of all reachable states of $(M, N)$, and each state in $R_K$ is a progress state.

(Only if) Assume that there is a positive integer $K$ such that each state in the $K$-reachable set $R_K$ of $(M, N)$ is a progress state. This implies that no state in $R_K$ is an overflow state, and the communication of $(M, N)$ is bounded by $K$. Therefore, the set $R$ of all reachable states of $(M, N) = R_K$, and the communication of $(M, N)$ is bounded, and free from deadlocks and unspecified receptions. ∎

From Theorem 1, the communication progress problem can be re-stated as follows: "Given $M$ and $N$, is the communication of network $(M, N)$ bounded, and free from deadlocks and unspecified receptions?" It is straightforward to show that a solution to this problem can solve the halting problem of Post machines (Manna, 1974). Hence this problem is undecidable.

THEOREM 2. *It is undecidable whether the communication of any network $(M, N)$ is bounded and free from deadlocks and unspecified receptions. (The undecidability holds even if $M$ and $N$ exchange message values and even if both machines are deterministic and have no mixed nodes.)* ▮

There are two approaches to bypass this negative result. First, identify special classes of communicating machines for which the problem is decidable. One example of this approach is discussed in the next section. The second approach is based on the observation that for most instances one is more interested in proving a positive answer to the problem. Therefore in Sections V and VI, we discuss two sets of sufficient conditions that ensure a positive answer for the communication progress problem.

## IV. ALTERNATING COMMUNICATING MACHINES

A communicating machine $M$ is called *alternating* iff each of its sending edges is followed by receiving edges only.

THEOREM 3. *The communication of any network of two alternating machines is bounded by two.*

*Proof.* Let $M$ and $N$ be two alternating machines. Figure 1 shows a directed graph $G$ that represents the reachable states of $(M, N)$. Each vertex in $G$ is labeled with a pair $(x, y)$, where $x$ and $y$ indicate the contents of the input channels to $M$ and $N$ respectively; in particular, each of $x$ and $y$ can have any of the following values:

$E$ to indicate that the channel is empty,

$g$ to indicate that the channel has one message, and

$gg$ to indicate that the channel has two messages.

Each arc in $G$ is labeled with a symbol that indicates one operation, or edge, executed by $M$ or $N$:

$s_M(s_N)$ indicates a sending edge executed by $M$ $(N)$.

$r_M(r_N)$ indicates a receiving edge executed by $M$ $(N)$.

Notice that since $M$ $(N)$ is alternating, then between any two successive arcs that are labeled $s_M$ $(s_N)$ there must exist an arc labeled $r_M$ $(r_N)$ in $G$. From $G$, the communication of $(M, N)$ is bounded by two. ▮

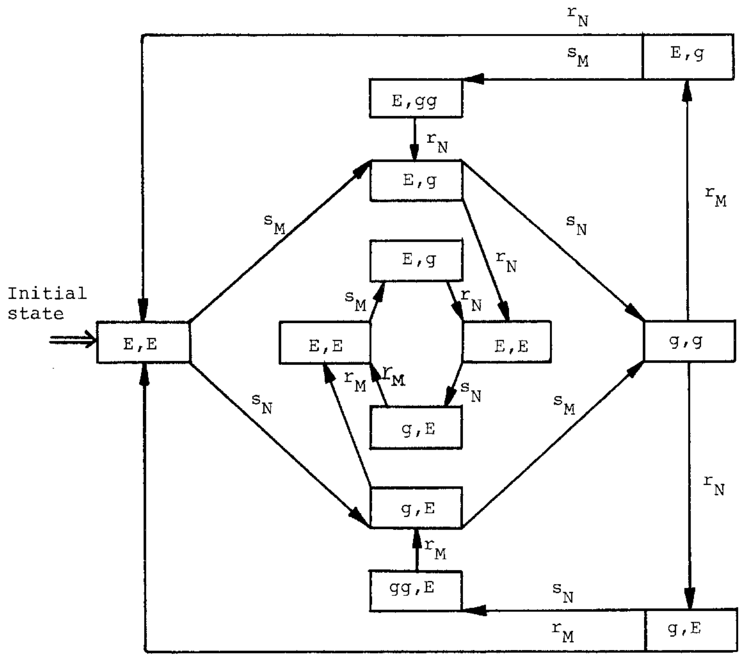


FIG. 1. Proof of Theorem 3.

COROLLARY 1. *The communication progress problem for any network $(M, N)$, where both $M$ and $N$ are alternating, can be decided in $O(mn|G|^4)$ time, where $m$ is the number of nodes in $M$, $n$ is the number of nodes in $N$, and $|G|$ is the number of messages exchanged between $M$ and $N$.*

*Proof.* From Theorem 3, any reachable state $[v, w, x, y]$ is such that $|x| \leqslant 2$ and $|y| \leqslant 2$. Thus, the number of reachable states is $O(mn|G|^4)$.

EXAMPLE 1. The alternating-bit protocol was proposed by Bartlet *et al.* (1969) to ensure reliable transmission of data messages from a sender to a receiver over a communication medium that can corrupt or lose transmitted messages. If a data message is corrupted or lost during transmission, or if its positive acknowledgment is corrupted or lost, then the data message is retransmitted. The retransmission is triggered by any of the following:

(i) The sender receives a negative acknowledgment.

(ii) The sender receives a corrupted message.

(iii) The sender waits a sufficient time period, called a timeout period, to receive a response; but no response is received indicating that either the original message or its response is lost.

When the receiver receives a data message, it should be able to detect whether it has received an identical copy of this message earlier. For this reason, the value of some bit in the sender is attached to each data message sent. So long as a data message is being retransmitted, the value of this bit remains fixed; but whenever a new data message is about to be sent, the value of this bit is altered (hence, the name "alternating-bit protocol").

Figure 2a shows a model of the alternating-bit protocol over a medium that corrupts messages. Machines $M_1$ and $N_1$ model the sender and the receiver, respectively. Instead of modeling the medium as a separate machine, the medium's effect is modeled as follows. Whenever a machine

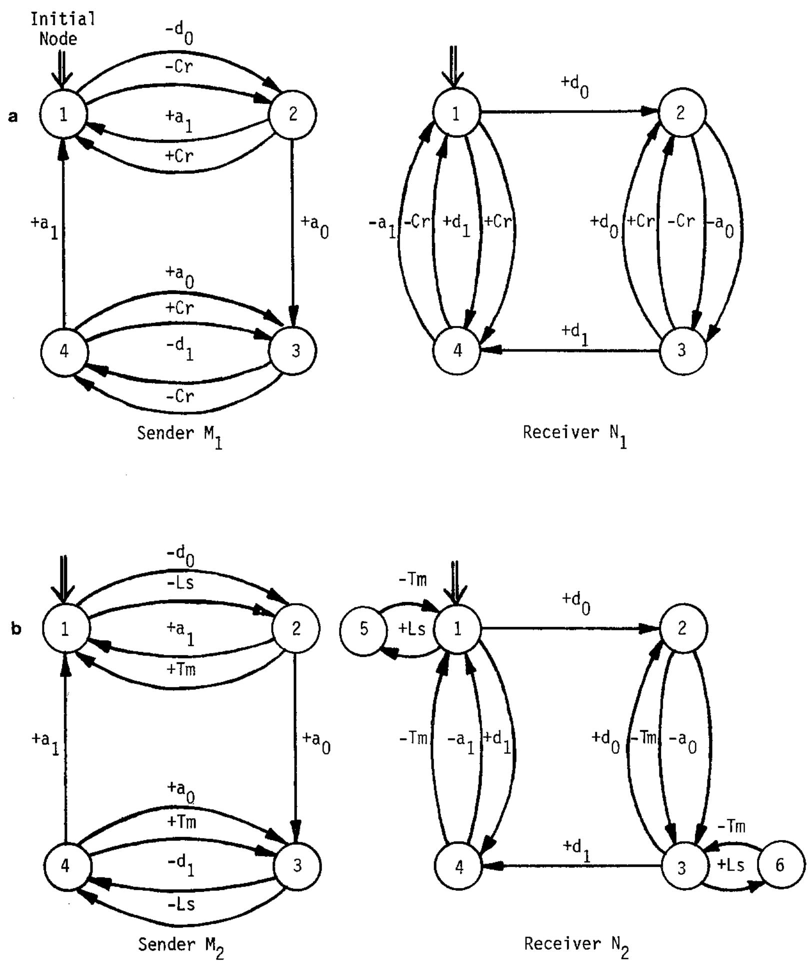


FIG. 2. An alternating-bit protocol (Notation: $-g$ means send($g$), $+g$ means receive($g$)): (a) modeling a medium that corrupts messages; (b) modeling a medium that loses messages.

($M_1$ or $N_1$) sends a message $g$, it either sends $g$ or sends a special message $Cr$ that denotes a corrupted message. The other exchanged messages between $M_1$ and $N_1$ have the following meanings.

$d_i$ ($i=0$, 1) denotes a data message with a bit of value "$i$" attached to it.

$a_i$ ($i=0$, 1) denotes a positive acknowledgment message for message "$d_i$", and a negative acknowledgment message for message "$d_{i+1\mathrm{mod}2}$."

Figure 2b shows a model of the alternating-bit protocol over a medium that loses messages. Machines $M_2$ and $N_2$ model the sender and the receiver, respectively. The medium's effect is modeled as follows:

(i) Whenever $M_2$ sends a data message $g$, it either sends $g$ or sends a special message $Ls$ that denotes a lost message. When $N_2$ receives the message $Ls$ it sends a special message $Tm$ that denotes a timeout message. When $M_2$ receives $Tm$ it sends another copy of the last sent data message.

(ii) Whenever $N_2$ sends a response message $g$, it either sends $g$ or sends the timeout message $Tm$ to simulate the loss of $g$ and force $M_2$ to resend the last sent data message.

To model the alternating-bit protocol over a medium that both corrupts and loses messages,

(i) join $M_1$ and $M_2$, by collapsing every two nodes with identical labels into one node, yielding a sender $M$, and

(ii) join $N_1$ and $N_2$, by collapsing every two nodes with identical labels into one node, yielding a receiver $N$.

Each of the machines $M_1$, $N_1$, $M_2$, $N_2$, $M$, and $N$ is alternating. Hence, the communications of the networks ($M_1$, $N_1$), ($M_2$, $N_2$), and ($M$, $N$) are all bounded (Theorem 3). It is straightforward to show that each of them is also free from deadlocks and unspecified receptions.

Two comments concerning this example are in order:

(i) The directed edge labeled $+a_1$ from node 2 to node 1, and the edge labeled $+a_0$ from node 4 to node 3 in $M_2$ will never be executed during the course of communication between $M_2$ and $N_2$. They have been added (redundantly) to simplify the argument for freedom from unspecified receptions. (See Example 3 in Section VI.)

(ii) In the above example, it is assumed that there is an upper bound on the time needed to send a message and receive its reply. (The timeout period is selected to be larger than this upper bound.) In the abscence of this upper bound, the timeout mechanism can cause the sender to resend an unbounded number of copies of the same data message, even if the original message was not corrupted or lost. In this case, the alternating-bit protocol can be modeled by a sender $M'$ and a receiver $N'$ as shown in Fig. 3. $M'$ is not alternating, hence the communication progress problem for $(M', N')$ cannot be decided by Theorem 3. On the other hand, it is straightforward to show that the communication of $(M', N')$ is *unbounded* (since $M'$ can reach a sending cycle), and *free from deadlocks* (since $M'$ has no receiving nodes), and *free from unspecified receptions* (since $N'$ expects, at each receiving node, to receive every possible message from $M'$.) ∎

In the next two sections, we discuss sufficient conditions that ensure a positive answer for the communication progress problem.

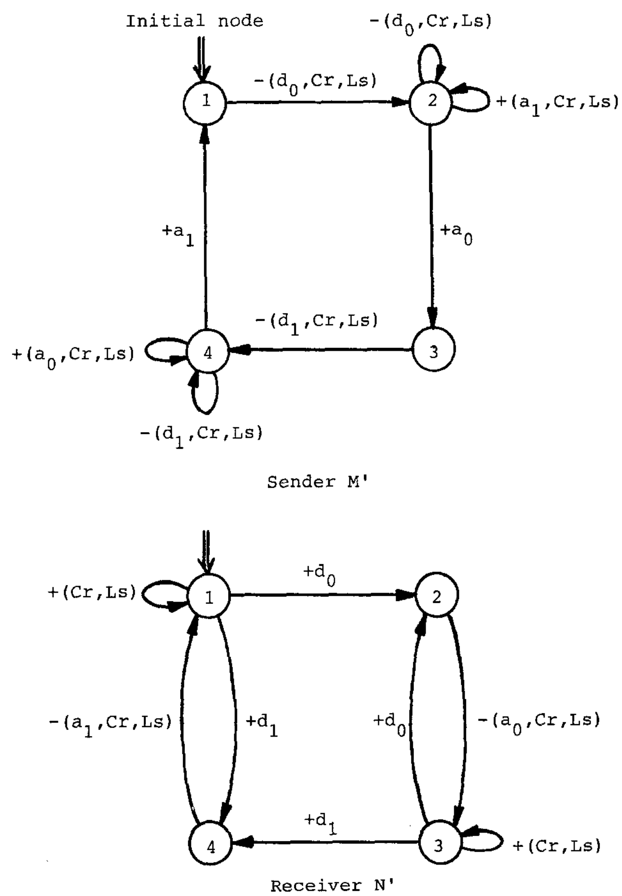


FIG. 3. An altenating-bit protocol without an upper bound on the round-trip time. (Notation: An edge from node $i$ to node $j$ labeled $-(g_1,..., g_r)$ represents $r$ edges from $i$ to $j$ labeled send$(g_1)$,..., send$(g_r)$. Similarly, $+(g_1,..., g_r)$ represents $r$ receiving edges.)

## V. Compatible Communication

Let $M$ and $N$ be two deterministic communicating machines that have no mixed nodes, and let $p$ and $q$ be two directed paths, of equal length, that start with the initial nodes in $M$ and $N$, respectively. Paths $p$ and $q$ are said to be *compatible paths* iff for $i = 1, 2,...,$ the ith edge in $p$ is labeled send($g$) (receive($g$)) and the $i$th edge in $q$ is labeled receive($g$) (send($g$)). The communication of $(M, N)$ is said to be *compatible* iff for any directed path $p$ that starts with the initial node in $M$, there exists *exactly one* directed path $q$ that starts with the initial node in $N$, and vice versa, such that $p$ and $q$ are compatible.

The reason for our interest in compatible communication is twofold. First, compatibility is a sufficient condition to ensure that the communication is deadlock-free and without unspecified receptions as we prove in Lemma 2. Second, it is decidable whether the communication of any network is compatible, as we prove in Lemma 3.

Lemma 2. *Let $M$ and $N$ be two deterministic communicating machines that have no mixed nodes. If the communication of $(M, N)$ is compatible then it is free from deadlocks and unspecified receptions.*

*Proof.* Assume that the communication of $(M, N)$ is compatible, and let $R$ be the set of all reachable states of $(M, N)$. We show by contradiction that no state $s$ in $R$ is a deadlock or an unspecified reception:

(i) *$s$ is a deadlock state.* Since $s$ is in $R$, there exist states $s_0,..., s_r$, such that $s_0$ is the initial state, $s_r = s$, and $s_i$ follows $s_{i-1}$ over some edge $e_i$ $(i = 1,..., r)$. The set of edges $\{e_i \mid i = 1,..., r \text{ and } e_i \text{ is in } M\}$ corresponds to a directed path $p$ which starts with the initial node in $M$. Similarly, the set of edges $\{e_i \mid i = 1,..., r \text{ and } e_i \text{ is in } N\}$ corresponds to a directed path $q$ which starts with the initial node in $N$. Since $s$ is a deadlock state, then $|p| = |q|$. There are two cases to consider:

(a) *$p$ and $q$ are compatible.* In this case, if path $p$ is extended in any way into $p'$ in $M$, then no directed path $q'$ which starts with the initial node in $N$ is compatible with $p'$. This contradicts the assumption that the communication between $M$ and $N$ is compatible.

(b) *$p$ and $q$ are not compatible.* Since the communication between $M$ and $N$ is compatible, there is a directed path $\tilde{q}$ that starts with the initial node in $N$ such that $p$ and $\tilde{q}$ are compatible. Clearly, $|q| = |p| = |\tilde{q}|$ and paths $q$ and $\tilde{q}$ are not identical. Let $e$ and $\tilde{e}$ be the first different edges in $q$ and $\tilde{q}$, respectively. Edges $e$ and $\tilde{e}$ have the same tail node; and they are either the $i$th sending edges or the $i$th receiving edges in their respective paths. Therefore, they correspond to the $i$th receiving edge or the $i$th sending edge of path $p$ in $M$; i.e., they have identical labels; this contradicts the fact that $N$ is deterministic.

(ii) *$s$ is an unspecified reception state.* A similar argument as in case (i) leads to a contradiction. ▮

LEMMA 3. *Whether the communication of a network $(M, N)$, where both $M$ and $N$ are deterministic and have no mixed nodes, is compatible, can be decided in $O(s^* \log s^* |G|)$ time where $s = \max(m, n)$, $m$ is the number of nodes in $M$, $n$ is the number of nodes in $N$, and $|G|$ is the number of messages exchanged between $M$ and $N$.*

*Proof.* Construct machine $\tilde{N}$ from $N$ by replacing each sending node by a receiving node and vice versa, and by replacing each label "send($g$)" by "receive($g$)" and vice versa. View machines $M$ and $\tilde{N}$ as two finite automata over the alphabet $\{\text{send}(g), \text{receive}(g) | g \text{ is in } G\}$; and assume that each node in $M$ or $\tilde{N}$ is an accepting state. Each path that starts with the initial node in machine $M$ ($\tilde{N}$) corresponds to exactly one word in the regular language $L(M)$ ($L(\tilde{N})$) accepted by the automaton $M$ ($\tilde{N}$). Since both machines are deterministic, then the converse is also true, namely each word in the language $L(M)$ ($L(\tilde{N})$) corresponds to exactly one path that starts with the initial node in machine $M$ ($\tilde{N}$)). Therefore, the communication between $M$ and $N$ is compatible iff $L(M) = L(\tilde{N})$, which can be decided in $O(s^* \log s^* |G|)$ time (Aho, Hopcroft, and Ullman, 1975). (This time is needed to construct minimum machines that are equivalent to $M$ and $N$. Deciding whether these minimum machines are equivalent takes negligible time.) ▮

From Lemma 2, compatibility guarantees freedom from deadlocks and unspecified receptions; however, it does not guarantee boundedness. Therefore, compatibility alone does not ensure a positive answer to the communication progress problem, and an additional condition is needed for that purpose.

THEOREM 4. *Let $M$ and $N$ be two deterministic communicating machines that have no mixed nodes. If the communication of $(M, N)$ is compatible and if each directed cycle in $M$ or $N$ has at least one sending and one receiving node, then the communication of $(M, N)$ is bounded and free from deadlocks and unspecified receptions.*

*Proof.* Since any directed cycle in $M$ or $N$ contains at least one sending and one receiving nodes, define $K$ to be the maximum number of successive nodes of the same type (sending or receiving) in $M$ or $N$. It is straightforward to show that any reachable state $s = [m, n, x, y]$ of $(M, N)$ satisfies the following three conditions:

(i) $s$ is not a deadlock state.

(ii) $s$ is not an unspecified reception state.

(iii) $|x| \leqslant K$ and $|y| \leqslant K$. ∎

EXAMPLE 2. Consider the two communicating machines $M$ and $N$ in Fig. 4. They model a simplified version of the data transfer phase in the binary synchronous protocol (Lam, 1983), where $M$ models a sender that sends a stream of "data blocks" to a receiver $N$, and for each sent block $M$ receives either a positive or negative acknowledgment from $N$. Each data block consists of a header field, a text field, and a 2-character check sum to detect corruption that may occur in the data block during transmission. The exchanged messages are as follows:

| | |
|---|---|
| *SOH* | is a start of header character, |
| *H* | is a header character, |
| *STX* | is a start of text character, |
| *T* | is a text character, |
| *ETX* | is an end of text character, |
| *BCC* | is a block check character, |
| *ACK* | is a positive acknowledgment message, and |
| *NACK* | is a negative acknowledgment message. |

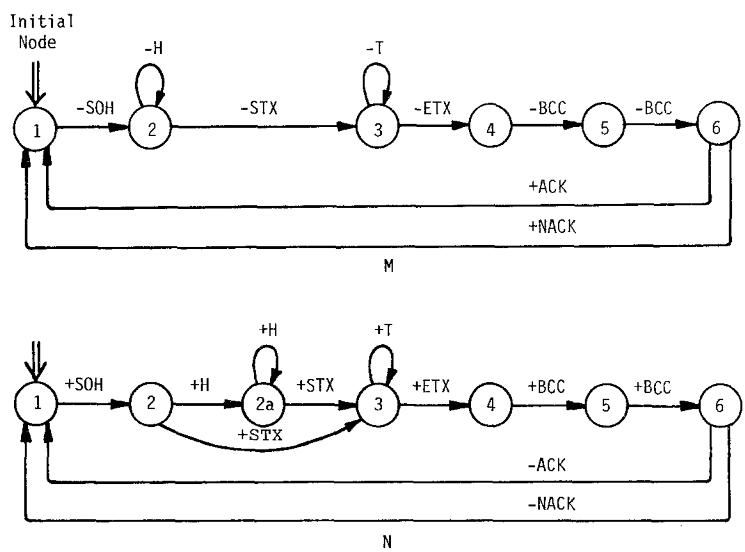


FIG. 4. Data transfer in the binary synchronous protocol. (Notation: $-g$ means send($g$), $+g$ means receive($g$).)

By inspection, the communication of $(M, N)$ is compatible; therefore it is free from deadlocks and unspecified receptions by Lemma 2. The communicitation of $(M, N)$ is unbounded. (This is because $M$ has two sending self-loops violating one of the two conditions of Theorem 4.) However, if there is an upper bound on the number of $H$ and $T$ characters allowed in each data block, then the conditions of Theorem 4 are satisfied, and the communication becomes bounded. (Most implementations of this protocol put an upper bound of 256 characters on the size of each data block, thus ensuring boundedness.) ∎

In the next section we discuss another set of conditions that ensures a positive answer for the communication progress problem.

## VI. Reducing the Number of Message Types

Let $M$ and $N$ be two communicating machines with the same set $G = \{g_1, \ldots, g_r\}$, $r \geqslant 2$, of messages. Let $M\langle g_2 : g_1 \rangle$ denote machine $M$ after replacing each $g_2$ label with a $g_1$ label, and let $N\langle g_2 : g_1 \rangle$ denote machine $N$ after replacing each $g_2$ label with a $g_1$ label. The two machines $M\langle g_2 : g_1 \rangle$ and $N\langle g_2 : g_1 \rangle$ exchange one less type of message, namely $g_2$, than $M$ and $N$. In this sense, the network $(M\langle g_2 : g_1 \rangle, N\langle g_2 : g_1 \rangle)$ is an "abstraction" of the network $(M, N)$.

Theorem 5. *If the communication of* $(M\langle g_2 : g_1 \rangle, N\langle g_2 : g_1 \rangle)$ *is deadlock-free (bounded), then the communication of* $(M, N)$ *is also deadlock-free (bounded).*

*Proof.* The theorem follows from the obervation that if $s$ is a reachable state of $(M, N)$, then $s\langle g_2 : g_1 \rangle$ is a reachable state of $(M\langle g_2 : g_1 \rangle, N\langle g_2 : g_1 \rangle)$, where $s\langle g_2 : g_1 \rangle$ is constructed from $s$ by replacing each occurrence of $g_2$ with an occurrence of $g_1$. ∎

Corollary 2. *Let* $M\langle g_1 \rangle$ *denote machine* $M$ *after replacing each message label with a* $g_1$ *label, and let* $N\langle g_1 \rangle$ *denote* $N$ *after replacing each message label with a* $g_1$ *label. If the communication of* $(M\langle g_1 \rangle, N\langle g_1 \rangle)$ *is deadlock-free (bounded), then the communication of* $(M, N)$ *is deadlock-free (bounded).*

*Proof.* $M\langle g_1 \rangle = ((M\langle g_2 : g_1 \rangle) \cdots)\ \langle g_r : g_1 \rangle$. Therefore, Corollary 2 follows from applying Theorem 5 $(r-1)$ times. ∎

The two machines $M\langle g_1 \rangle$ and $N\langle g_1 \rangle$ exchange one type of message (namely $g_1$); hence it is decidable (Cunha and Maibaum, 1981; Yu and Gouda, 1982; Yu and Gouda, 1983) whether their communication is deadlock-free and/or bounded. This fact along with Corollary 2 suggest the following methodology to prove that the communication progress problem for a given network $(M, N)$ with a message set $G$ has a positive answer:

(i) Use the polynomial decidability algorithm in Yu and Gouda (1983) to prove that the communication of $(M\langle g_1\rangle, N\langle g_1\rangle)$ is both deadlock-free and bounded. This guarantees that the communication of $(M, N)$ is both deadlock-free and bounded (by Corollary 2).

(ii) To guarantee that the communication of $(M, N)$ is also free from unspecified receptions, it is sufficient to ensure that each receiving node in $M$ or $N$ has an outgoing edge labeled receive($g$), for each message "$g$" in set $G$.

EXAMPLE 3. Consider the two communicating machines $M_1$ and $N_1$ in Fig. 2a. (Recall that they model the alternating-bit protocol over a medium that corrupts messages.) If each message label in $M_1$ or $N_1$ is replaced with a $d_0$ label, then the resulting two machines $M_1\langle d_0\rangle$ and $N_1\langle d_0\rangle$ are as shown in Fig. 5. It is straightforward to show that the communication of $(M_1\langle d_0\rangle, N_1\langle d_0\rangle)$ is both deadlock-free and bounded. By Corollary 2, this ensures that the communication of the original network of $(M_1, N_1)$ is both deadlock-free and bounded. Moreover, since each receiving node in $M_1$ $(N_1)$ has an outgoing edge labeled receive($g$) for every message $g$ sent by $N_1$ $(M_1)$, the communication of $(M_1, N_1)$ is also free from unspecified receptions. This completes the proof that the communication progress problem has a positive answer for the two machines $M_1$ and $N_1$. (A similar argument can be used to show that the communication progress problem has a positive answer for the two machines $M_2$ and $N_2$ in Fig. 2b) ▮

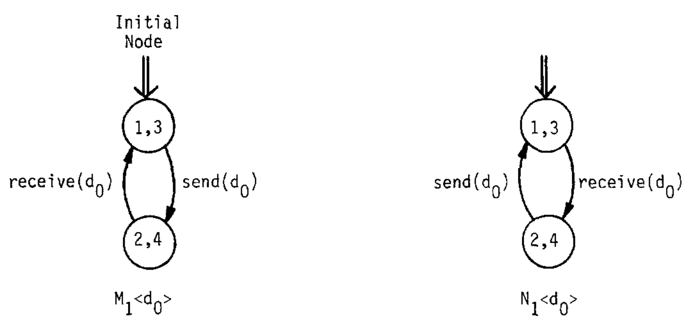


FIG. 5. Abstracting the alternating-bit protocol using two machines that exchange one type of message.

## VII. Concluding Remarks

We have shown that the communication progress problem for two communicating finite state machines is undecidable in general, and discussed two approaches to bypass this rather negative result. The first approach is to characterize special classes of machines for which the problem becomes decidable. The second approach is to find sufficient conditions that ensure a positive answer to the problem (since this is our goal in most cases).

There is a third approach to bypass this negative result, namely "synthesis." Instead of starting with two machines and trying to decide whether their communication will progress indefinitely, one can start with only one machine, then synthesize the second machine such tha the communication is guaranteed to progress indefinitely. For more details about this approach, the reader is referred to Gouda and Yu (1984b).

## Acknowledgments

The authors are thankful to Professor A. Meyer for his encouragement and to the referees whose suggestions have greatly improved the presentation.

Received September 28, 1982; accepted March 14, 1985

## References

Aho, A. V., Hopcroft, J. E., and Ullman, J. D. (1975), "*The Design and Analysis of Computer Algorithms*," Addison–Wesley, Reading, Mass.

Bartlet, K. A., Scantlebury, R. A., and Wilkinson, P. T. (1969), A note on reliable full-duplex transmission over half-duplex lines, *Comm. ACM* **12**, No. 5, 260–261.

Bochmann, G. V. (1978), Finite state description of communication protocols, *Comput. Networks* **2**, 361–371.

Brand, D. and Zafiropulo, P. (1983), On communicating finite-state machines, *J. Assoc. Comput. Mach.* **30**, No. 2, 323–342.

Cunha, P. R., and Maibaum, T. S. (1981), A synchronization calculus for message oriented programming *in* "*Proc.* 2nd *International Conf. on Distributed Computing Systems*," April 1981, pp. 433–445.

Danthine, A. (1980), Protocol representation with finite state models, *IEEE Trans. Comm.* **COM-28**, No. 4, 632–643.

Gouda, M. G., and Yu, Y. T. (1984a), Protocol validation by maximal progress state exploration, *IEEE Trans. Comm.* **COM-32**, No. 1, 94–97.

Gouda, M. G., and Yu, Y. T. (1984b), Synthesis of Communicating Machines with Guaranteed Progress, *IEEE Trans. Comm.* **COM-32**, No. 7, 846–855.

Lam, S. S. (1983), Data link control procedures, *in* "*Computer Communications*, Vol. I., Principles" (W. Chou, Ed.), pp. 81–113, Prentice–Hall, Englewood Cliffs, N.J.

Manna, Z. (1974), "*Mathematical Theory of Computation*," McGraw–Hill, New York.

SUNSHINE, C. A. (1981), "Formal Modeling of Communication Protocols," *Research Report* 81–89, USC/Inform. Sci-Institute.

WEST, C. H. (1978), An automated technique of communication protocol validation, *IEEE Trans. Comm.* **COM-26**, 1271–1275.

YU, Y. T., AND GOUDA, M. G. (1982), Deadlock detection for a class of communicating finite state machines, *IEEE Trans. Comm.* **COM-30**, No. 12, 2514–2518.

YU, Y. T., AND GOUDA, M. G. (1983), Unboundedness detection for a class of communicating finite state machines, *Inform. Process. Lett.* **17**, 235–240.

ZAFIROPULO, P., *et al.* (1980), Towards analyzing and synthesizing protocols, *IEEE Trans. Comm.* **COM-28**, No. 4, 651–661.