DEADLOCK DETECTION FOR A CLASS OF COMMUNICATING FINITE STATE MACHINES

Yao-Tin Yu and Mohamed G. Gouda

Department of Computer Sciences
University of Texas at Autin
Austin, Texas 78712

TR-193 Feb 1982

ABSTRACT

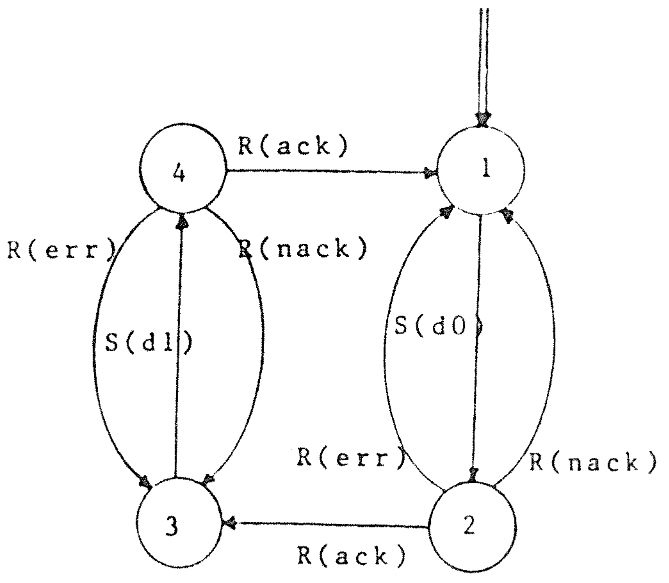
Let M and N be two nonterminating communicating finite state machines where each of them sends one class of messages to the other. We develope a polynomial algorithm to detect whether M and N can reach a deadlock. The time complexity of the algorithm is $O(m^3 n^3)$ and its space is $O(mn)$ where m and n are the numbers of states in M and N respectively.

Keywords: Communicating finite state machines, communication deadlocks, deadlock detection.

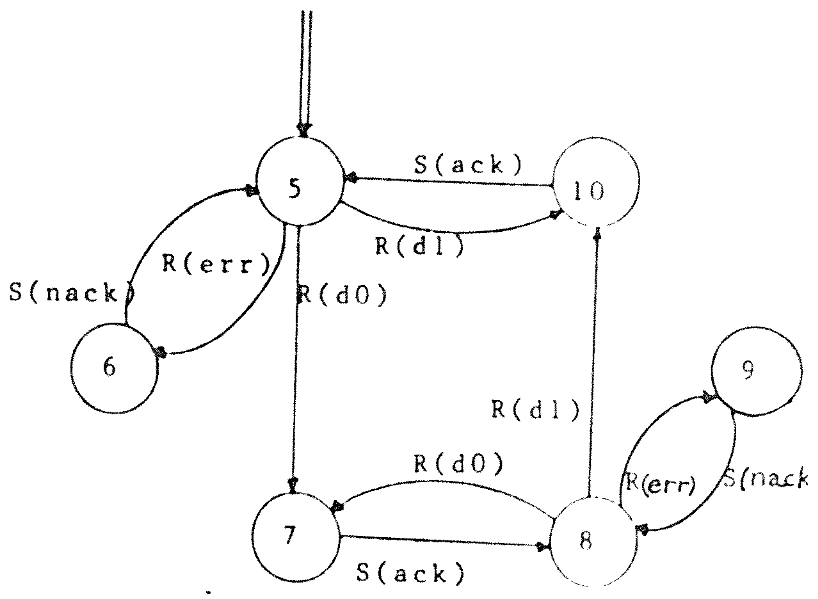
I. INTRODUCTION

The model of communicating finite state machines is an abstraction of sequential processes which communicate exclusively by exchanging messages. It can be used to specify [4],[8], analyze [1],[2],[3] and synthesize [5],[7], [9] communication protocols. In [2], it is shown that the problem of whether two communicating finite state machines can reach a deadlock state is undecidable in general. The problem is shown [3] to be decidable if each machine sends one type of messages to the other machine. In this paper, we consider this same restricted class of machines and give an efficient decidability algorithm which is polynomial in both time and space.

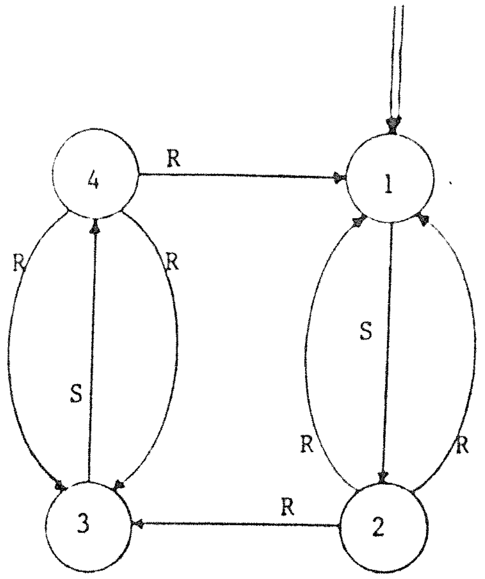
There are practical motivations to consider this restricted class of communicating machines. Let $\mathbf{M}$ and $\mathbf{N}$ be two communicating finite state machines which exchange many types of messages. If $\mathbf{M}$ and $\mathbf{N}$ are abstracted by two machines M and N, each of which sends one type of messages to the other; and if the communication between M and N is shown to be deadlock-free, then the communication between $\mathbf{M}$ and $\mathbf{N}$ is also deadlock-free. (The reverse is not necessarily true.) As an example, Figures 1a and 1b show two communicating finite state machines $\mathbf{M}$ and $\mathbf{N}$ which implement an alternating-bit protocol [7] to transmit data messages from $\mathbf{M}$ to $\mathbf{N}$ through a medium where messages can be corrupted during transmission. Informally, a directed edge in $\mathbf{M}$ or $\mathbf{N}$ corresponds to either a sending or a receiving operation; an edge corresponding to a sending (or a receiving) operation is labelled



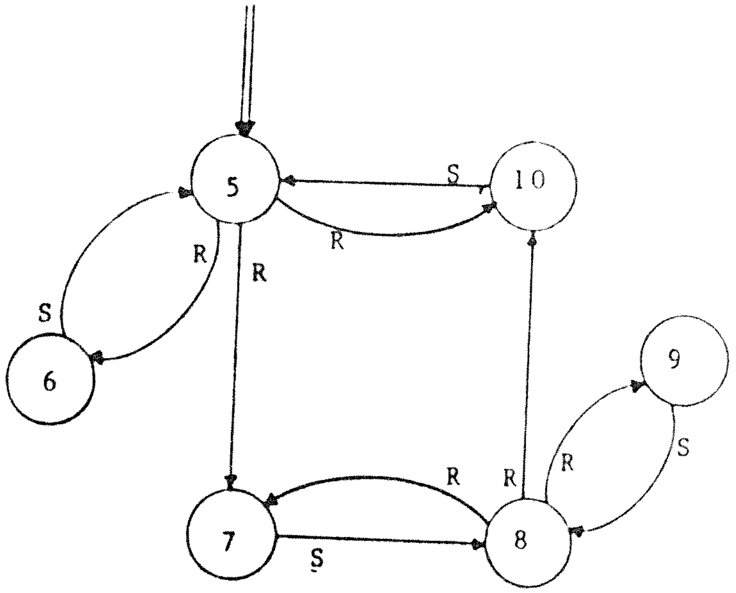
(a) $\bar{M}$ sends two types of data messages "d0" and "d1"



(b) $\bar{N}$ sends two types of messages "ack" and "nack"



(c) M: an abstraction of $\bar{M}$.



(d) N: an abstraction of $\bar{N}$.

Figure 1: An alternating_bit protocol

S(m) (or R(m) respectively) for some message type m. The two machines $\bar{M}$ and $\bar{N}$ can be abstracted by M and N in Figures 1c and 1d respectively. Using ALGORITHM 1, discussed below, it can be shown that the communication between M and N is deadlock-free; hence, the communication between $\bar{M}$ and $\bar{N}$ is also deadlock-free.

Another application for this restricted class of communcating finite state machines is in modeling some classes of self-timing VLSI arrays. This leads to proving (or disproving) efficiently that these arrays are free of communication deadlocks [6].

II. COMMUNICATING FINITE STATE MACHINES

A communicating finite state machine M is a directed labelled graph where nodes are called *states*; each state in M is either *sending* or *receiving state*. Each edge is labelled either "S" or "R" depending on whether its tail state is a sending or a receivng state. An edge labelled "S" (or "R") is called a *sending* (or *receiving* respectively) *edge*. One of the states in M is identified as its *initial state*; and all the states in M are reachable by directed paths from its initial state. For convience, we assume that M is "nonterminating", i.e., each state in M must have at least one output edge.

Let M an N be two cimmunicating finite state machines. A *golbal state* of M and N is an ordered tuple with four components [v,w,x,y] where v and w are two states in M and N respectively and x and y are two non-negative integers. Informally, a global state [v,w,x,y] implies that if the execution of M has reached state v, and the execution of N has reached state w, then the imput channels of M and N have x and y messages respectively.

The <u>initial global state</u> of M and N is $[v_0, w_0, 0, 0]$ where v_0 and w_0 are the initial states of M and N respecively.

A golbal state [v,w,x,y] of M and N is a <u>deadlock state</u> iff (i) both v and w are receiving states and (ii) x=y=0.

Let s=[v,w,x,y] be a global state of M and N, and let e be an output edge of state v or w. A global state s′ is said to <u>follow</u> s <u>over</u> e, denoted s--e-->s′, if the following four conditions are satisfied:

1. <u>If</u> e is a sending edge from v to v′ in M <u>then</u> s′=[v′,w,x,y+1].
2. <u>If</u> e is a sending edge from w to w′ in N <u>then</u> s′=[v,w′,x+1,y].
3. <u>If</u> e is a receiving edge from v to v′ in M <u>then</u> $x \geq 1$ and s′=[v′,w,x-1,y].
4. <u>If</u> e is a receiving edge from w to w′ in N <u>then</u> $y \geq 1$ and s′=[v,w′,x,y-1].

Let s and s′ be two global states of M and N, s′ <u>follows</u> s if there is a directed edge e in M or N such that s--e-->s′.

Let s and s′ be two global states of M and N. s′ is <u>reachable from</u> s if s=s′ or there exist states $s_1, s_2, \ldots, s_r$ such that $s=s_1$, $s'=s_r$, and s_{i+1} follows s_i for i=1,...,r-1. The set of all global states which are reachable from the initial global state is called <u>reachable set</u>.

Consider the following problem. "Given two communicating finite state machines M and N whose reachable set is R, does R have a deadlock state?" In [3], this problem is shown to be decidable.

The decidability algorithm creates a bounded tree called the synchronization tree for M and N which is a finite representation for the reachable set; the complexity of this algorithm is yet to be determined. In this paper, we give a more efficient algorithm to solve the problem, and show that this algorithm is polynomial in time and space.

III. A SOLUTION

A global state[v,w,x,y] of M and N is called _fair_ iff x=y. Obviously, the initial global state of M and N is fair; also any deadlock state is fair.

Let s=[v,w,x,x] be a fair global state of M and N; and let d and e be two output edges of states v and w respectively. A fair global state s′ is said to _fairly follow_ s _over_ d and e, denoted s==d,e==>s′, if the following four conditions are satisfied:

1. _If_ d is a sending edge from state v to state v′ in M, and e is a sending edge from state w to state w′ in N _then_ s′=[v′,w′,x+1,x+1].

2. _if_ d is a receiving edge from state v to state v′ in M, and e is a sending edge from state w to state w′ in N _then_ s′=[v′,w′,x,x].

3. _If_ d is a sending edge from state v to state v′ in M, and e is a receiving edge from state w to state w′ in N _then_ s′=[v′,w′,x,x].

4. _If_ d is a receiving edge from state v to state v′ in M, and e is a receiving edge from state w to state w′ in N _then_ $x \geq 1$ and s′=[v′,w′,x-1,x-1].

Let s and s′ be two fair global states of M and N. s′ _fairly follows_ s if there exist two directed edges d and e in M and N respectively such that s==d,e==>s′.

Let s and s′ be two fair global states of M and N. s′ is <u>fairly reachable from</u> s if s=s′ or there exist fair global states $s_1, s_2, \ldots, s_r$ such that $s=s_1$, $s'=s_r$, and s_{i+1} fairly follows s_i for $i=1,\ldots,r-1$. The set of all fair global states which are fairly reachable from the initial global state is called the <u>fair set</u> of M and N.

For the next lemmas and theorems, let M and N be two communicating finite state machines; and let R and F be their reachable and fair sets respectively.

Lemma 1: Let s and s′ be two fair global states of M and N. If s′ fairly follows s then s′ is reachable from s.

Proof: Assume that s=[v,w,x,x] and s′=[v′,w′,x′,x′] and that s==d,e==>s′. There are four cases to consider:

(i) Both d and e are sending edges:

Since s′ fairly follows s, then x′=x+1,

[v,w,x,x]--d-->[v′,w,x,x+1] and

[v′,w,x,x+1]--e-->[v′,w′,x+1,x+1]. So s′ is reachable from s.

(ii) d is a sending edge and e is a receiving edge:

Since s′ fairly follows s, then x′=x, [v,w,x,x]--d-->[v′,w,x,x+1] and [v′,w,x,x+1]--e-->[v′,w′,x,x]. So s′ is reachable from s.

(iii) d is a receiving edge and e is a sending edge:

Using a similar argument as in (ii), it can be shown that s′ is reachable from s.

(iv) Both d and e are receiving edges:

Since s′ fairly follows s, then $x \geq 1$, $x' = x-1$, $[v,w,x,x] \text{--}d\text{-->} [v',w,x-1,x]$ and $[v',w,x-1,x] \text{--}e\text{-->} [v',w',x-1,x-1]$. So s′ is reachable from s. []

Theorem 1: Any deadlock state in R must also be in F, and vice versa.

Proof:

Part(i) Any deadlock state in F is in R:

Assume that s is a deadlock state in F. Then there exist fair states $s_0,\ldots,s_r$, such that s_0 is the intial global state, $s_r = s$ and s_{i+1} fairly follows s_i, $i=0,\ldots,r-1$. From Lemma 1, if s_{i+1} fairly follows s_i then s_{i+1} is reachable from s_i. Therefore s is reachable from the initial global state, i.e., s is in R.

Part(ii) Any deadlock state in R is in F:

Let s be a deadlock state in R. Then there are states ,$s_0,\ldots,s_r$, in R, such that s_0 is the initial global state, $s_r = s$, and $s_i \text{--}f_{i+1}\text{-->} s_{i+1}$, $i=0,\ldots,r-1$, where f_i is an edge in M or N. Partition the sequence of edges $F=f_1,\ldots,f_r$, into two subsequences of edges $D=d_1,\ldots,d_m$ and $E=e_1,\ldots,e_{m'}$ where d_i (or e_i) is an edge in M (or N respectively). Since s_r is a deadlock state, the number of sending (or receiving) edges in D is equal to the number of receiving (or sending respectively) edges in E. So, $m=m'=r/2$.

Let d_i be from v_i to v_{i+1} in M, and e_i be from w_i to w_{i+1} in N, $i=1,\ldots,m$. Construct a sequence of fair states $s'_0,\ldots,s'_m$ by the following rules: (i) s'_0 is the initial global state, and (ii) $s'_i=[v_i,w_i,x_i,x_i]$ where $x_i=x_{i-1}+1$ if d_i and e_i are sending edges, $x_i=x_{i-1}-1$ if d_i and e_i are receiving edges, and $x_i=x_{i-1}$ otherwise. In order to prove that s'_m is fairly reachable from s'_0, it is sufficient to prove that each x_i is nonnegative, $i=1,...,m$. Since for $i=0,...,r-1$, s_i is not a deadlock state, so the ith sending edge in D must appear in F before the ith receiving edge in E. Also the ith sending edge in E must appear in F before the ith receiving edge in D. This guarantees that for $i=1,...,m$, $x_i \geq 0$.

[]

Theorem 2: Let $s=[v,w,x,x]$ and $s'=[v,w,y,y]$ be two states in F where $x>y$. If there exists a deadlock state which is fairly reachable from s, then there exists a deadlock state which is fairly reachable from s'.

Proof: Let s_d be a deadlock state which is fairly reachable from s. Then there is a sequence of fair states $s_1,..,s_r$ such that $s=s_1$, $s_r=s_d$ and s_{i+1} fairly follows s_i, $i=1,..,r-1$. Assume that $s_i=[v_i,w_i,x_i,x_i]$ where $i=1,..,r$, and $x_1=x$ and $x_r=0$. Since x_i's decrease from x to 0, there should be a smallest integer k, $0\leq k\leq r$, such that $x_k=t$ and $x_{k+1}=t-1$ where $t=x-y$ is greater than zero. This implies that v_k and w_k are both receiving states, and that for each x_i $(i=1,...,k)$, $x_i \geq t$. Define $s'_i=[v_i,w_i,x_i-t,x_i-t]$, $i=1,..,k$. Hence $s'_1=s'$, s'_{i+1} fairly follows s'_i, $i=1,..,k-1$, and

$s' = [v_k, w_k, 0, 0]$ is a deadlock state. Therefore a deadlock state is fairly reachable from s'. []

From Theorem 1, there is no need to generate all the reachable states to check the existance of deadlock states; the fair states are sufficient. From Theorem 2, there is no need to generate all the fair states either; this is because after a fair state $[v,w,y,y]$ is generated then any fair state $[v,w,x,x]$ where $x \geq y$ need not be generated. These remarks motivate the following algorithm.

ALGORITHM 1:

Input: Two communicating finite state machines, M and N, and two sets, called OPEN and CLOSE, of pairs (v,w) where v and w are states of M and N respectively. (Initially OPEN and CLOSE are empty)

Output: _If_ CLOSE has a pair (v,w) _then_ [v,w,0,0] is deadlock state of M and N _else_ no deadlock state of M and N is reachable.

Steps:

```
1. for any state v of M and state w of N do MSGNO(v,w):=~
   endfor

2. if $v_0$ and $w_0$ are the initial states of M and N respectively
       then MSGNO($v_0$,$w_0$):=0; add the pair ($v_0$,$w_0$) to set OPEN
       endif

3. while OPEN is not empty do
   a. Remove one element from OPEN; let this element be (v,w);
   b. Generate all fair states which fairly follow state
        s=[v,w,MSGNO(v,w),MSGNO(v,w)];
   c. if there are no such followers of s
          then add (v,w) to CLOSE;
          else
          for each follower $[v_i,w_i,x_i,x_i]$ of s do
              if $MSGNO(v_i,w_i)>x_i$
                 then $MSGNO(v_i,w_i):=x_i$;
                     if $(v_i,w_i)$ is not currently in OPEN
                       then add it to OPEN;
                       endif
                 endif
          endfor
       endif
   endwhile                                                     []
```

IV. CORRECTNESS AND COMPLEXITY

The correctness of ALGORITHM 1 and its polynomial complexity are established by the following lemmas and theorem.

Lemma 2: If CLOSE has a pair (v,w) at the termination of ALGORITHM 1 then [v,w,0,0] is a deadlock state.

Proof: If a pair (v,w) is in CLOSE at the termination of ALGORITHM 1, then a fair state s=[v,w,x,x], fairly reachable from the initial global state of M and N, has been generated by ALGORITHM 1 and that s has no followers. This implies that both v and w are receiving states and that x=0;i.e., s is a deadlock state. []

Lemma 3: Let MSGNO(v,w)=m, and MSGNO(v′,w′)=m′ at the termination of ALGORITHM 1. If fair state [v′,w′,x′,x′] fairly follows fair state [v,w,x,x] and $x \geq m$ then either [v,w,m,m] is a deadlock state or $x' \geq m'$.

Proof: From step 3.c., since MSGNO(v,w)=m at the termination of ALGORITHM 1, the fair state [v,w,m,m] must have been generated in ALGORITHM 1, and the pair (v,w) must have been added to OPEN during the algorithm execution. If [v,w,m,m] is a deadlock state, then the lemma is correct. Assume that [v,w,m,m] is not a deadlock state; then all the fair states which fairly follow [v,w,m,m] must have been generated in the algorithm. Since [v′,w′,x′,x′] fairly follows [v,w,x,x], a fair state [v′,w′,z′,z′], where z′=m−(x−x′), must have been generated as a follower of [v,w,m,m]. Since at the algorithm termination MSGNO(v′,w′)=m′, then $z' \geq m'$. From z′=m−(x−x′), $z' \geq m'$ and $x \geq m$, so $x' \geq m'$. []

Lemma 4: If F has a deadlock state then CLOSE is not empty at the termination of ALGORITHM 1.

Proof: Assume that s is a deadlock state in F. Then there are fair states $s_0,\ldots,s_r$ such that s_0 is the initial global state of M and N, s_r=s, and s_{i+1} fairly follows s_i, i=0,...,r−1. Let $s_i=[v_i,w_i,x_i,x_i]$ and let $\mathrm{MSGNO}(v_i,w_i)=m_i$ at the termination of ALGORITHM 1. If for each i, i=0,...,r, $[v_i,w_i,m_i,m_i]$ is not a deadlock state, then from Lemma 3, $x_i \geq m_i$, i=1,...,r, since $x_0 \geq m_0=0$. Since $x_r=0$ then $m_r=0$. Therefore $[v_r,w_r,0,0]$ must have been generated in ALGORITHM 1; and because it has no followers, the pair (v_r,w_r) must have been added to CLOSE. On the other hand, if for some j, j=0,...,r , $[v_j,w_j,m_j,m_j]$ is a deadlock state (i.e., $m_j=0$) then (v_j,w_j) must have been added to CLOSE. Therefore if F has some deadlock state then CLOSE must not be empty at the termination of ALGORITHM 1. []

Lemma 5: ALGORITHM 1 terminates

Proof: It is sufficient to prove that the number of pairs added to OPEN is finite; this is equivalent to proving the following three assertions:

1. If a pair (v,w) is added to OPEN, then there exists a nonnegative integer x such that [v,w,x,x] is generated in ALGORITHM 1.
2. For any fair state [v,w,x,x] generated in ALGORITHM 1, the pair (v,w) is added to OPEN at most once.
3. If [v,w,x,x] is generated in ALGORITHM 1, then $x<mn$, where m and n are the number of states in M and N respectively.

Assertion(1): From step 3.b., every global state generated in ALGORITHM 1 is a fair state. And from step 3.c., a pair (v,w) is added to OPEN only when [v,w,x,x] has been generated, where x is a nonnegative integer.

Assertion(2): Assume that a fair state [v,w,x,x] has been generated in ALGORITHM 1 and that the pair (v,w) is added to OPEN at instant t. At this instant, MSGNO(v,w)=x. Since the value of MSGNO(v,w) is monotonically decreasing (from step 3.c.), then at any subsequent instant $\text{MSGNO}(v,w)\leq x$. If at a later instant [v,w,x,x] is generated then (v,w) will not be added to OPEN (from step 3.c.).

Assertion(3): Assume that $s=[v,w,x,x]$ is generated in the algorithm. Then there must exist fair states, $s_i=[v_i,w_i,x_i,x_i]$, $i=0,\ldots,r$, generated in ALGORITHM 1 such that

1. s_0 is the initial global state of M and N,
2. $s_r=s$,
3. s_{i+1} fairly follows s_i, $i=0,\ldots,r-1$, and
4. If for some j and k, $0\leq j<k\leq r$, $(v_j,w_j)=(v_k,w_k)$ then $x_j>x_k$ (otherwise s cannot be generated in ALGORITHM 1 by this sequence).

Let P_i be the number of distinct pairs in $\{(v_0,w_0),\ldots,(v_i,w_i)\}$, we prove that $x_i<P_i$ which implies that $x_i<mn$, $i=0,\ldots,r-1$ and $x<mn$. The proof is by induction on i.

- Initial step(i=0): $x_0=0$ and $P_0=1$; then $x_0<P_0$.
- Induction hypothesis($i\leq k$): Assume that $x_k<P_k$.
- Induction step(i=k+1):
  If $(v_{k+1},w_{k+1})=(v_j,w_j)$ for some $j,0\leq j<k$; then $x_{k+1}<x_j<P_j<P_{k+1}$ by condition 4 above. Otherwise, let $(v_{K+1},w_{k+1})\neq(v_j,w_j)$ for any j, $j=0,\ldots,k$;i.e., $P_{k+1}=P_k+1$. Since x_{K+1} can be greater than x_k by 1 and $x_k<p_k$, then $x_{k+1}<p_{k+1}$. []

Lemmas 2,4, and 5 establish the correctness of ALGORITHM 1; the following theorem establishes its complexity.

Theorem 3: The time and space complexities of ALGORITHM 1 are $O(m^3n^3)$ and $O(mn)$ respectively.

Proof: (i) Time complexity

The number of followers for any global state is $O(mn)$. Since for each [v,w,x,x], the pair (v,w) is added to OPEN at most once, and since $x \leq mn$ by lemma 4, then the loop in step3 of ALGORITHM 1 is executed an $O(m^2n^2)$ times. So the time complexity of ALGOTITHM 1 is $O(m^3n^3)$.

Notice that in order to achieve this time complexity OPEN should be implemented as a boolean array "open" such that open[v,w]=true iff (v,w) is in OPEN, and a "counter" whose value is the current number of elements in OPEN.

(ii) Space complexity

It is trivial to show that the spaces needed for MSGNO, CLOSE, and OPEN are $O(mn)$. []

V. CONCLUDING REMARKS

Detecting communication deadlocks for more than two communicating machines where each machine sends one type of messages to any other machine, is still an open problem in general. In some cases, however, it can be shown that r communicating machines ($r>2$) can reach a deadlock state iff any two of them can reach a deadlock state. In these cases, ALGORITHM 1 can be used to efficiently detect communication deadlocks for the r machines. As an example, ALGORITHM 1 has been used to verify that a number of VLSI arrays are free of communication deadlocks [6].

In this paper, it is assumed that the outputs of any state in a communicating finite state machine are all sending edges or all receiving edges. Detecting communication deadlocks for two mmachines where any state can have both sending and receiving outputs is still an open problem.

REFERENCES

[1] G.V. Bochmann,"Finite state description of communication protocols," Computer Networks, Vol 2.1978, pp.361-372.

[2] D. Brand, and P. Zafiropulo, "On communicating finite-state machines," IBM Research Report, RZ 1053 (#37725) Jan 81.

[3] P.R.F. Cunha, T.S.E. Maibaum, "A synchronization calculus for message oriented programming," Research Rep. CS-80-43, Dept. of Computer Science, Univ. of Waterloo, Sep. 1980.

[4] A. Danthine, "Protocol representation with finite state models," IEEE Trans on Comm., Vol.COM_28, No.4, April 1980, pp.632-643.

[5] M.G. Gouda, and Y. Yu, "Designing deadlock-free and bounded communication protocols," Tech. Rep-173, Dept. of Computer Sciences, Univ. of Texas at Austin, June 1981.

[6] M.G. Gouda, "On proving that a VLSI array is free of communication deadlocks," in preparation.

[7] P. Merlin, and G.V. Bochmann, "On the construction of communication protocols and module specification," Pub. 352, Dept. dinformatique de recherche op′ erationnelle, Universit′e de Montreal, Jan 1980.

[8] C.A. Sunshine, "Formal modeling of communication protocols," USC/Inform. Sc. Institute, Research Report 81-89, March 1981.

[9] P. Zafiropulo, et al., "Towards analyzing and synthesizing protocols," IEEE Trans. on Comm., Vol. COM-28, No.4, April 1980, pp.651-661.