DECIDING PROGRESS FOR A CLASS OF COMMUNICATING FINITE STATE MACHINES[1]

Louis E. Rosier and Mohamed G. Gouda

TR-83-22 October 1983

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

[1]This research was supported in part by the University Research Institute, the University of Texas at Austin and the IBM Corporation.

Abstract

Consider networks of two communicating finite state machines that exchange messages over two unbounded, one-directional, FIFO channels. The communication progress problem for these networks is to decide whether the communication of an arbitrary network is bounded and/or free from deadlocks and unspecified receptions. This problem is known to be undecidable, even if the two machines exchange only two types of messages. However, the problem becomes decidable if the two machines exchange only one type of message. In this paper, we sharpen this boundary between the decidable and undecidable cases, and examine the class of networks where only *one* of the two machines sends one type of message (the communication in the other direction is not constrained). We show that for this class boundedness can be decided, and each reachable deadlock or unspecified reception state can be identified. Furthermore, we show that these decision problems are nondeterministic logspace complete and thus boundedness, freedom from deadlocks and unspecified receptions can all be decided in polynomial time. Lastly, using similar techniques, we are able to give optimal (providing PTIME $\neq$ PSPACE) decision procedures for the general case of two machines, where one of the two channels is known a priori to be bounded.

1. INTRODUCTION

Many communication protocols can be modeled as a network of two finite state machines that communicate by exchanging messages over two unbounded, one-directional, FIFO channels [1,2,5,11,15,16,17]. (Generalizations of this model permit any number of communicating finite state machines, each pair of which communicate as above.) Each machine has a finite number of states (called nodes) and state transitions (called edges). Each state transition of a machine is accompanied by either sending one message to the output channel of the machine or receiving one message from the input channel of the machine. (Formal definitions are presented later.)

As an example, consider the network of two communicating finite state machines M and N, shown in Figure 1. M is a sender that sends data messages to a receiver N; N responds by sending acknowledgements to M. The communication proceeds as follows:

1. M starts by sending a sequence of two data messages, $data_1$ followed by $data_2$.
2. N responds by sending an acknowledgement. Based on this acknowledgement, M performs one of two actions: if the acknowledgement is totally negative or totally positive, M sends a sequence of two messages, $data_1$ followed by $data_2$. (If the acknowledgement is totally negative, then this sequence is a repeat of the previously sent sequence; otherwise, it is the next sequence of data messages.) If the acknowledgement is partially positive (indicating that $data_1$ has been received correctly and that only $data_2$ needs to be retransmitted), then M sends $data_2$.
3. It is possible that after N receives $data_1$, it discovers that this message needs to be retransmitted. In this case, N sends the acknowledgement right away without waiting for $data_2$. If M receives this acknowledgement before sending $data_2$, then it must go back and resend $data_1$.
4. For i=0,1, each $data_i$ message consists of the following characters:
   one S_i (for start of text) character,
   zero or more T_i (for text) characters, and
   one E_i (for end of text) character.

Notice that M sends six types of messages namely S_1, T_1, E_1, S_2, T_2, and E_2 while N sends one type of message namely A (for acknowledgement).

One advantage of modeling a communication protocol by a network of communicating finite state machines is to detect many protocol design errors. Three of these design errors which have been discussed extensively in the literature are unboundedness, deadlocks, and unspecified receptions (cf. [1,2,4,15,16-18]). In this paper we give nondeterministic logspace algorithms to detect these three design errors (plus others) for the class of networks where one of the two machines sends only one type of message (the communication in the other direction is not constrained). (Note that the network shown in Figure 1 is in this class.) Moreover, we are able to show that each reachable unspecified reception (deadlock) state can be identified, within this bound. Furthermore, we show that these decision problems are nondeterministic logspace complete and thus boundedness, freedom from deadlocks and unspecified receptions can all be decided in polynomial time[3].

These results are important for two reasons:

i. It has been shown earlier[2,5] that detecting any of these design errors is undecidable for the class of networks in which each machine is allowed to send two types of messages. It has also been shown earlier [4,8,9,10,12,16,17] that detecting any of these design errors is decidable for

networks where each machine sends one type of message. Therefore, the current result fills the gap in a positive way between these two undecidable/decidable results. Moreover, by showing that the given decision procedures are polynomial, the current result generalizes the result in [16] where only deadlock detection is shown to be polynomial for a special class of networks where the two machines exchange one type of message.

ii. The current result can be used to prove that a general network of two machines M and N that exchange any number of messages is free from the above design errors. First, abstract M and N into two machines M_1 and N_1, where M_1 sends only one type of message, by considering each message sent by M (and received by N) as the same. (The communication in the other direction is left as is.) Then use the decision procedure in this paper to prove that M_1 and N_1 are free from the above design errors. Second, abstract M and N into two machines M_2 and N_2, where N_2 sends only one type of message. Then use the decision procedure in this paper to prove that M_2 and N_2 are free from the above design errors. It is straightforward to show that if M_1 and N_1 are free from the above design errors and if M_2 and N_2 are free from these errors, then the original machines M and N are also free from these same errors. The converse need not be true, of course. An example to illustrate the abstraction procedure is given in Figure 2.

In [2], a partial procedure was given to determine, for a network consisting of an arbitrary number of communicating finite state machines, whether any unspecified reception or deadlock states were reachable (as well as whether the network had any nonexecutable receptions). The procedure in [2] was shown to terminate for the class of communication networks consisting of two finite state machines in which the communication was bounded in at least one of the channels. The algorithm can also be modified slightly in order to determine whether the remaining channel is bounded. Thus, for the case of two machines the above three design errors can be detected by a decision procedure if one of the two channels is known a priori to be bounded. The time complexity of the procedure is not discussed in [2], however, one can show, that it must be a function not only of the size of the network, but also of the known channel bound. Using the techniques presented in our paper, we give a different decision procedure for this same problem and show that the time complexity of our procedure is the best that can be achieved, unless PTIME=PSPACE.

Although in this paper we discuss only the detection of three design errors (namely unboundness, deadlocks, and unspecified receptions), it can be shown that other design errors (e.g. nonexecutable receptions, and stable states [2,18]) can be detected as well using the same decision procedures.

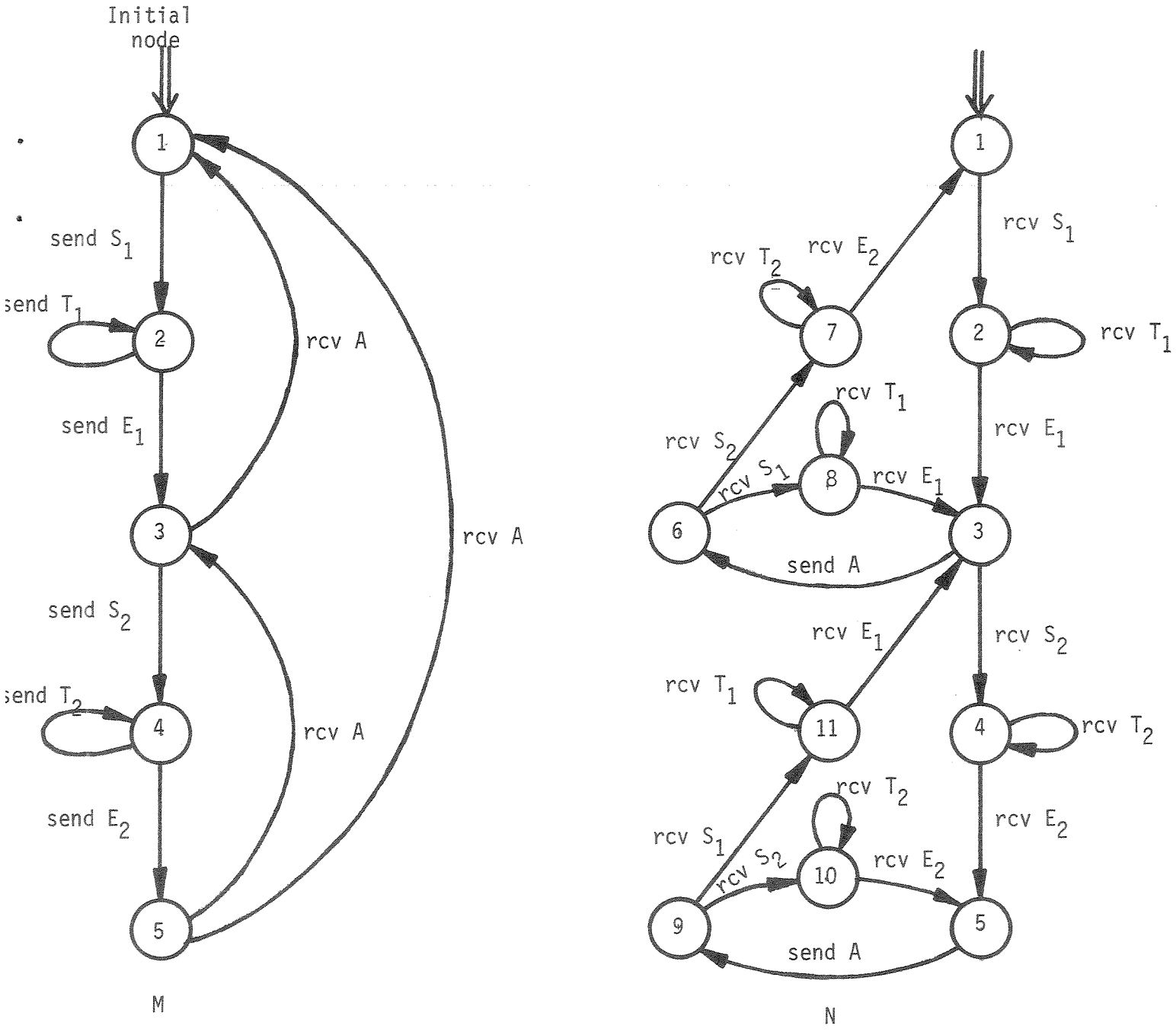


Figure 1. An example of two communicating finite state machines

2. COMMUNICATING FINITE STATE MACHINES

A *communicating finite state machine* M is a directed labelled graph with two types of edges, namely *sending* and *receiving edges*. A sending (receiving) edge is labelled send(g) (receive(g)) for some *message* g in a finite set G of messages. A node in M whose outgoing edges are all sending (all receiving) edges is called a *sending* (*receiving*) *node*; a node in M whose outgoing edges include both sending and receiving edges is called a *mixed node*. One of the nodes in M is identified as its *initial node*; each node is reachable by a directed path from the initial node. A node which a directed edge indents from (or to) is called the *source* (*destination*) node of the edge.

Let M and N be two communicating finite state machines with the same set G of messages; the pair (M,N) is called a *network* of M and N. A *state* of network (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] denotes 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 network (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 are satisfied:

i. e is a sending edge, labelled send(g), from v to v' in M, and s'=[v',w,x,y'], where y'=y.g (".” is the string concatenation operator).

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

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

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

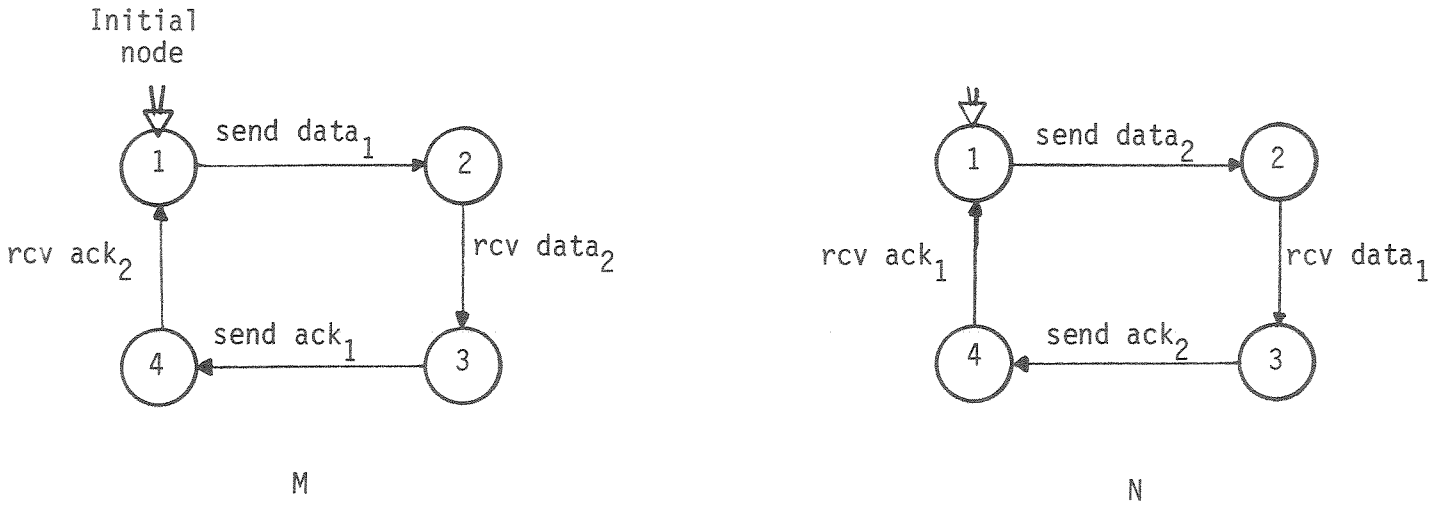
Let R be a set of states of the network (M,N), and let s and s' be two states of (M,N). Then s' is *reachable from* s (*within R*) iff s=s' or there exist states $s_1,...,s_r$ such that $s=s_1$, $s'=s_r$ and s_{i+1} follows s_i for i=1,...,r-1 (and $s_1,...,s_r$ are in R).

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

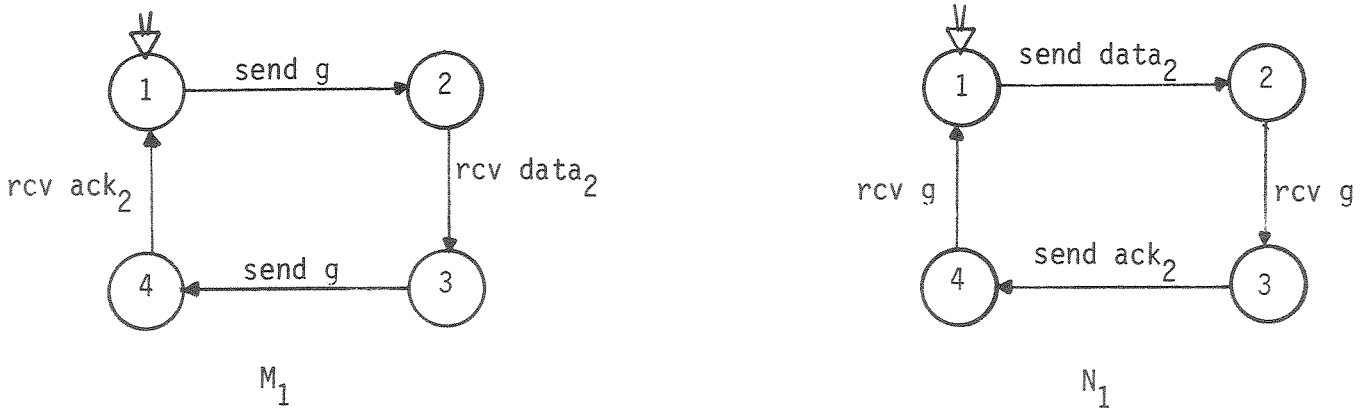
The reachability set of network (M,N) is the set of all reachable states of (M,N). Note that if R is the reachability set of (M,N), then a state of the network is reachable iff it is reachable within R.

A *finite computation* of a network is a sequence of states $s_0,...,s_r$ in which s_0 is the initial state of the network and s_{i+1} follows from s_i, $0\leq i\leq r-1$. An *infinite computation* is such a sequence but with infinite length.

The communication of network (M,N) is said to be *bounded by* k, where k is a nonnegative integer, iff for every reachable state [v,w,x,y] of (M,N), $|x|\leq k$ and $|y|\leq k$ where |x| is the number of messages in x. The communication is said to be *bounded* iff it is bounded by some nonnegative integer k; otherwise it is *unbounded*.



(a) The original machines



(b) The first abstraction

(c) The second abstraction

Figure 2. An example to illustrate the abstraction procedure

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

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

i. $x=g_1.g_2. \ldots .g_k$ $(k\geq 1)$; and v is a receiving node and none of its outgoing edges is labelled receive(g_1).

ii. $y=g_1.g_2. \ldots .g_k$ $(k\geq 1)$; and w is a receiving node and none of its outgoing edges is labelled receive(g_1).

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

A state of network (M,N) is a *nonprogress* state if it is either a deadlock state or an unspecified reception state.

3. DECISION PROBLEMS

In this section we show that polynomial time decision procedures can be constructed, to determine whether the communication is unbounded and whether the network can reach a deadlock and/or unspecified reception state, for the following two classes of communication networks:

C_1: The class of networks consisting of two finite state machines in which each machine sends only one type of message to the other machine, i.e. G only contains a single message.

C_2: The class of networks consisting of two finite state machines in which at least *one* of the machines sends only one type of message to the other machine, i.e. each edge labelled by a send (in one of the machines) mentions the same message of G. (Note that C_1 is properly contained within C_2.)

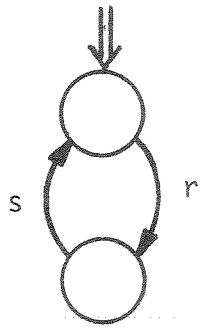
In fact we prove a slightly stronger result, which is that all three of these problems are nondeterministic logspace complete for both C_1 and C_2. Thus it follows from results in [3] that these problems are solvable in polynomial time. (See [6] for motivations and definitions of nondeterministic logspace hard, nondeterministic logspace complete, etc. See also [7,14].) This generalizes results in [16,17], where such problems were considered for subclasses of C_1. Our approach is to first show that, to determine whether a network is unbounded and/or whether it's reachability set contains a deadlock state, is nondeterministic logspace hard for the class C_1. (Note that unspecified reception states do not exist for C_1 networks.) Then we show that all three problems can be decided in nondeterministic logspace for the class C_2. To determine, whether the reachability set of an arbitrary C_2 network contains an unspecified reception state, is also nondeterministic logspace hard. Although we do not explicitly show this, it follows using a similar construction as the one presented in Theorem 1.

We need the following problem which is known to be complete for nondeterministic logspace with respect to logspace reductions [13]. The "graph reachability problem" is, given a directed graph with vertices $\{1,...,n\}$, determine if there is a path from 1 to n in G. We are now ready to show the following easy result.

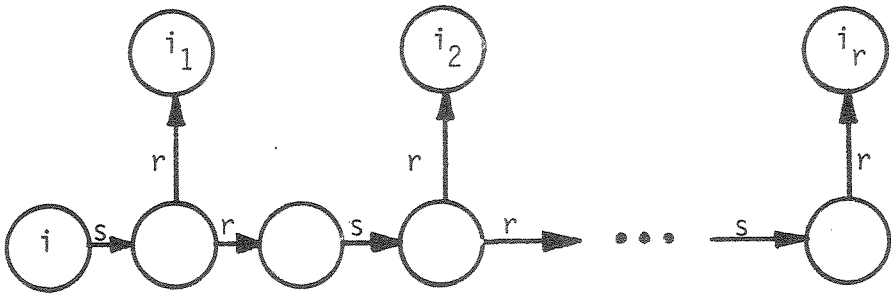
Theorem 1: To determine, for an arbitrary C_1 network, whether the network is unbounded and/or whether the network's reachability set contains a deadlock state, is nondeterministic logspace hard.

Proof: We show how to construct a logspace transducer W that, when given an arbitrary directed graph $G=(V,E)$ (let $V=\{1,...,n\}$), produces as output a network (M,N) in C_1, whose communication is unbounded (has a deadlock) iff there is a path from 1 to n in G. The construction of M is trivial and is shown in figure 3a. The construction of N depends on the graph G. N has nodes labelled 1 through n that correspond to the vertices of G, in addition to other nodes. The node labelled 1 is the start node. Suppose that vertex i of G, for $i<n$, has directed edges to vertices $i_1,...,i_r$. Then the portion of N corresponding to vertex i of G, is shown in figure 3b. The portion of N corresponding to vertex n of G, is shown in figure 3c (3d), for the case when we are concerned with whether the network is unbounded (whether the network has a deadlock). The reader can easily see that G has a directed path from 1 to n iff the network (M,N) is unbounded (has a deadlock). The details of the construction of the transducer W are left to the reader. []

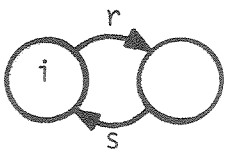
The reader should note that the network constructed in Theorem 1 has no mixed nodes. Thus, the result also holds for the subclass of C_1 considered in [16]. In order to prove the results concerning the class C_2, we need some preliminary work. Let (M,N) be an arbitrary network in the class C_2. Without



a. Machine M.



b. The portion of N for node i, $1 \leq i < n$, with $r \geq 1$ outgoing edges.



b. The portion of N for node i, $1 \leq i < n$, without outgoing edges.

c. The portion of N for node n if we are considering boundedness.

d. The portion of N for node n if we are considering freedom of deadlocks.

Figure 3. Network (M,N) for the proof of Theorem 1.

loss of generality we assume that N sends only one type of message to M. Let R(M,N) (hereafter denoted by R) be the reachability set of the network. Define R'(M,N)={s| s in R and if s=[v,w,x,y] then $|y| \leq 1$} (denoted simply as R'), i.e. R' is the set of reachable network states in which the input channel to N either contains zero or one message. We now prove the following lemma about R'.

Lemma 1: R'={the set of states that are reachable within R'}.

Proof: Clearly {the set of states reachable within R'} $\subseteq$ R'. Consequently we need only show the reverse. Let s be in R'. Since s is reachable by definition within R, there exists a computation of the network $s_0,...,s_r$ such that s_0 is the initial state of the network and s_r=s. Consider separately, but in order, the sequence of moves made by M and N in this computation. These moves are interleaved to form the computation of the network indicated by the sequence $s_0,...,s_r$. One can construct another computation of the network utilizing exactly the same two sequences of moves, but interleaving them in a different way such that the same state s is eventually reached and each intermediate state is in R'. This can be done by always choosing to execute the next move of N (if one remains to be executed), instead of the next move of M, whenever the input channel of N contains a message and the next move of M, to be executed, sends a message to N. The details of this construction are left to the reader. []

Corollaries 1,2, and 3 now follow almost immediately from Lemma 1.

Corollary 1: If a deadlock state s is in R, then s is in R'.

Corollary 2: If an unspecified reception state s=[v,w,x,g.y] is in R, then there exists an unspecified reception state s'=[v',w,x',g] in R', for some node v' in M and some string x' over G.

Corollary 3: The channel from N to M is unbounded iff R' is infinite.

Proof: If the channel from N to M is unbounded, there is an infinite computation of the network in which the contents of the input channel of M become arbitrarily large. Reordering the moves of M and N, in this computation, as prescribed in Lemma 1, does not decrease (and in fact may increase) the contents of this channel at any point in the computation. Consequently, R' is infinite. []

From Corollaries 1 and 2, we find that to determine whether the network (M,N) has a reachable deadlock or unspecified reception state, it is sufficient to consider only those states which are in R'. Likewise to determine if the channel from N to M is unbounded, it is sufficient to determine if R' is infinite. We are almost ready to prove our next theorem, but in order to do so, we require the following definitions.

A deterministic one counter automaton (doca) is a deterministic pushdown automaton (see [6]) with a stack alphabet of just one symbol (besides the bottom-of-stack marker). It is the case here that we can restrict ourselves to those doca's that do not utilize ϵ-moves and where each move can change the stack height by at most one. Formally, a doca W is a 7-tuple $<Q,\Sigma,\Gamma,\delta,q_0,Z_0,F>$, where Q is a set of states, Σ the input alphabet, $\Gamma=\{Z_0,B\}$ is the stack alphabet (note that the bottom-of-stack marker Z_0 can neither be written nor erased), δ: $Q \times \Sigma \times \Gamma \rightarrow Q \times \{-1,0,+1\}$ is the transition function, q_0 is the initial state, Z_0 is the bottom-of-stack marker and F is the set of accepting states.

A configuration c of W is described by a pair (q,i) where $q \epsilon Q$ and i is a nonnegative integer representing the contents of the counter (or stack). A computation $c \stackrel{\alpha}{\rightarrow} c'$ is a sequence of moves specified by the transition function δ, that leads from c to c', and which in the process causes W to read the input word α in Σ^*. (Note that in our case the number of moves in the computation is $|\alpha|$, the length of the

string α.) A computation is positive if no intermediate configuration in the computation has a zero counter. We are now ready to prove our next theorem.

Theorem 2: For the class C_2, the following three decision problems are solvable in nondeterministic logspace:

1. Decide whether the reachability set of the network contains a deadlock state.
2. Decide whether the reachability set of the network contains an unspecified reception state.
3. Decide whether the channel through which only one type of message can be sent, is unbounded.

Proof: Let (M,N) over the message set G, be in C_2. Let v_0 (w_0) be the initial node in M (N). Without loss of generality, we assume that N only sends one type of message to M. Let R and R' denote the sets of reachable states as indicated above. Let $\ell=(|M|+|N|)*(|G|+1)$. Notice that the total number of possible moves that can be made from any network state is no more than ℓ. Hence, the possible moves of the network can, for any network state, be indexed by $1,...,\ell$. One can now easily construct from (M,N) a doca $W=<Q,\Sigma,\Gamma,\delta,q_0,Z_0,F>$ with input alphabet $\Sigma=\{1,...,\ell\}$, whose reachable configurations coincide with the states in R'. The construction of W is as follows. $Q=\{r,a,a'\} \cup \{(v,w,g) \mid v$ (w) is a node in M (N) and g is either a message in G or E (indicating that the input channel of M is empty)$\}$. The initial state of W is (v_0,w_0,E). The state a' is the accept state, i.e. $F=\{a'\}$. The state r is a reject state, i.e. no moves can be made from the state r. The contents of the counter are used to store the number of messages contained in the input channel of M. The moves of W (as long as W has not already determined that the input will be rejected or accepted) are essentially the moves of the network, restricted such that a state in R-R' is not encountered. Hence, in state (v,w,g) with input i, W will simulate the ith move of the network (from this state) if such a move exists; otherwise W will reject the input. Whenever $g \neq E$ and the ith move from (v,w,g) is a move of M that sends a message to N, then W also rejects the input. When simulating a move, W updates the internal state to reflect the new state of the network, as well as the counter contents if the move of the network altered the channel contents of M's input channel. If W ascertains that it is in a deadlock state or unspecified reception state, depending on whether we are interested in the first or second decision problem, then W on any input can enter state a. (Note that W must actually be in a deadlock (unspecified reception) state for this transition to occur.) W, in state a, empties the counter and enters state a', the accepting state. (The first definition of acceptance should be used for the third decision problem.) We leave it to the reader to establish that there is a logspace transducer that when given a network (M,N) in C_2, can produce the desired doca W. Clearly then R' contains a deadlock (unspecified reception) state iff W accepts some input string.

Now we need that W accepts some input string iff it accepts some short input string (i.e. polynomial in |Q|, the number of states in Q). We now show that this short string need be no longer that $2*|Q|^4$. (Notice that the pumping lemma for pushdown automata would yield the existence of a short string, but it's length may be exponential in |Q|.) Call a computation $c_1,...,c_2$, in which c_1 and c_2 are in the same state of Q, a "loop". Now suppose that there is an accepting computation of W. We will show that there is another accepting computation, in which the counter contents never exceed $2*|Q|^3$. Consider now any accepting computation of W (e.g. the one represented in figure 4). Let b be a point (time) in the computation, in which the counter value exceeds $2*|Q|^3$. Let point a be the most recent time before b in which the counter value was exactly $|Q|^3$. Let c be the next time, after time b, in which the counter value is again $|Q|^3$.

Define intervals (a_i,b_i), (b'_i,c_i), $0 \leq i \leq |Q|-1$ where a_i = the last time before b that the counter value was $|Q|^3+i*|Q|$, b_i = the first time after a_i that the counter value was $|Q|^3+(i+1)*|Q|$, b'_i = the last time

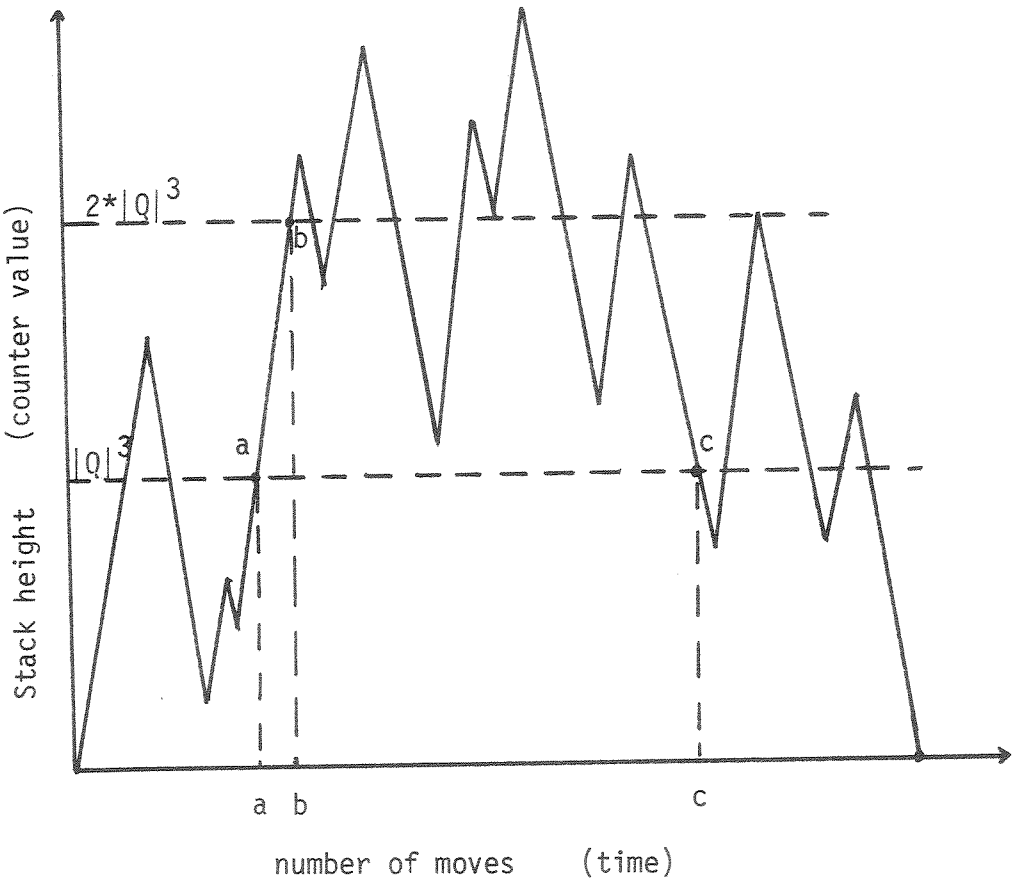


Figure 4. An accepting computation of W.

before c that the counter value was $|Q|^3+(i+1)*|Q|$, and c_i = the first time after b'_i that the counter value was $|Q|^3+i*|Q|$. Now between times a_i and b_i (b'_i and c_i), W must have executed some loop, say ℓ_i (ℓ'_i) in which there was a gain (loss) in the counter value of between 1 and $|Q|$. This must be the case, since during the interval (a_i, b_i), the counter gained $|Q|$ in value and at least $|Q|$ steps were executed.

Now there are $|Q|^2$ such loops between points a and b. Hence there must exist a k, $1\leq k\leq|Q|$, such that there are at least $|Q|$ loops with a gain in the counter contents of k. Likewise, there must exist a j, $1\leq j\leq|Q|$, such that there are at least $|Q|$ loops with a loss of j. Let u,v be such that k*u = the lowest common multiple of k and j = j*v ($1\leq u,v\leq|Q|$). Let $d_i \xrightarrow{\alpha_i} d'_i$, $1\leq i\leq u$, be the first u loops in $\{\ell_0,\ldots,\ell_{|Q|^2-1}\}$ that have a gain of k. Let $e_i \xrightarrow{\gamma_i} e'_i$, $1\leq i\leq v$, be the first v loops in $\{\ell'_0,\ldots,\ell'_{|Q|^2-1}\}$ that have a loss of j.

Now from the given computation, we can create a shorter computation by removing the α_i's ($1\leq i\leq u$) and the γ_i's ($1\leq i\leq v$) from the appropriate positions of the input, so as to avoid executing the associated loops. Note that the resulting computation is also an accepting computation, since the total increase in the counter contents caused by the α_i's is exactly offset by the total loss caused by the γ_i's. This process of shortening the computation can then be repeated until there is no time during the resulting computation in which the counter value exceeds $2*|Q|^3$.

Consequently, if there is an accepting computation we can find an accepting computation in which no configuration is repeated and in which the counter value never exceeds $2*|Q|^3$. Hence, the length of such a computation cannot exceed $2*|Q|^4$. Thus, whether a doca accepts some input string, can be decided in nondeterministic logspace.

We now turn our attention to the third problem. Clearly, R' is infinite iff the number of distinct reachable configurations of W is infinite. This can only be the case if there exists a computation of W: $c_0,\ldots,c_n$ where $c_i=(q_i,h_i)$, $0\leq i\leq n$ and $h_n\geq|q|$. To see this, suppose that $c_0,\ldots,c_n$ is the shortest such sequence. Let m be the largest nonnegative integer less than n such that $h_m=0$. Clearly, $n-m\geq|Q|$. Hence there must exist i and j, $m<i<j\leq n$ such that $q_i=q_j$. Let i and j be the least such i and j. Thus $j-m\leq|Q|$. Suppose $h_j\leq h_i$. Then one can construct a shorter such computation: $c_0,\ldots,c_i,c'_j,\ldots,c'_n$, where

$$c'_{j+k}=(q_{j+k},(h_{j+k}+(h_i-h_j)))$$

for $1\leq k\leq n-j$. This cannot happen since $h_i-h_j\geq 0$. Thus it must be the case that $h_j>h_i$. Now then in a similar fashion as before one can "pump" the loop determined by the computation $c_i,\ldots,c_j$ to produce an arbitrarily large counter value in some computation of W. Notice that in the above computation that $j-m\leq|Q|$. Since the counter in configuration c_0 through c_m is no greater than $|Q|$ (and each configuration is distinct), we have $m\leq|Q|^2$. Thus in the above computation $j\leq 2*|Q|^2$. This then yields an easy way to show that the third problem can be decided in nondeterministic logspace. Such an algorithm would nondeterministically look for a computation: $c_0,\ldots,c_m,\ldots,c_n$, where $n\leq 2*|Q|^2$, $c_i=(q_i,h_i)$ and where,

1. $c_m,\ldots,c_n$ is a positive computation and
2. $q_m=q_n$ and $h_n>h_m$.

Clearly this can be done in nondeterministic logspace. []

It is easy to see from the proof of the previous theorem the following:

Corollary 4: The detection of the specific deadlock and unspecified reception states can be accomplished in nondeterministic logspace.

Corollary 5: Let (M,N) over the message set G be in C_2. If N only sends one type of message to M, then M's input channel is bounded iff it is bounded by $2*(|M|*|N|*(|G|+1)+3)^2$.

Now we turn our attention to deciding for an arbitrary network (M,N) in C_2, where the channel constrained to a single type of message is bounded, whether the remaining channel is bounded.

Let (M,N) be in C_2 and let the channel from N to M (the channel constrained by a single type of message) be bounded by k. Let |M|=m and |N|=n. Then we claim that N's input channel is bounded iff it is bounded by n*(k+1). We prove this by showing that in any computation of the network, in which (at some instant) N's input channel exceeds n*(k+1), that M during the computation traversed a cycle in which all edges were labelled by sends. Consider such a computation of the network, $s_0,...,s_\ell$ where $s_\ell=[u,v,x,y]$ and $|x|\geq n*(k+1)$. One can construct from this computation another computation $s'_0,...,s'_\ell$ by interleaving the identical sequences of moves made by M and N in such a way as to always execute next a move of N (if one remains to be executed) instead of the next move of M, unless the next move of N is a read move and the input channel of N is empty. Clearly, $s_\ell=s'_\ell$. Also there must be an $i<\ell-(n*(k+1))$ such that each move in the computation from s_i to s_ℓ utilizes a move from M and $s_i=[u',v',E,y']$. In this computation there are at least n*(k+1) messages sent to N, but at most k messages to read. Consequently, M must have traversed a cycle in which all edges were labelled by sends. Thus from the discussion above and the proof of Theorem 2, we have the following lemma.

Lemma 2: Let (M,N) over the message set G be in C_2. Let |M|=m, |N|=n and |G|=k. Let the channel from N to M be constrained to a single type of message. Then the communication of (M,N) is bounded iff M's input channel is bounded by $2*(m*n*(k+1)+3)^2$ and N's input channel is bounded by $n*(2*(m*n*(k+1)+3)^2+1)$.

Clearly then from Theorem 2 and Lemma 2 we have:

Theorem 3: For the class C_2, the problem of deciding whether the communication is bounded, is solvable in nondeterministic logspace.

The reader should note that Lemma 2 can be generalized somewhat for the class of networks consisting of two finite state machines where one channel is (known to be) bounded (no other restriction is necessary). An algorithm to detect freedom from deadlocks and unspecified receptions for this class of networks was presented in [2]. Let (M,N) be such a network over G. Let k be the bound on M's input channel. Then N's input channel is either bounded by (k+1)*(|N|) or there is a computation of the network in which M traverses a cycle in which each edge is labelled by a send. Thus one can construct a deterministic ϵ-free finite state automaton (see [6]) with $O(|M|*|N|*(|G|+1)^{k+1})$ states, that accepts some input string iff the network has a reachable deadlock (unspecified reception) state. The finite state automaton remembers at most one message in N's input channel, as it simulates a move of N (instead of M), whenever the next move of M is a send and the channel is not empty. This provides an easier proof of the decidability results presented in [2].

As a result then, we have the following theorem.

Theorem 4. Let C_3 be the class of networks over G ($|G|\geq 2$) consisting of two finite state machines where one channel is known to be bounded. Then boundedness (of the remaining channel), freedom from deadlocks and unspecified receptions is decidable for C_3. Furthermore, if (M,N) is such a network where the known channel bound is k, the algorithm runs in space $O(k*\log(|M|*|N|))$.

Lastly, we show that the above problem cannot be solved in time polynomial in |M|, |N| and k unless PSPACE=PTIME. To show this we provide a reduction from the acceptance problem for (nondeterministic) linear bounded automata, which is well known to be PSPACE-complete. (See [6] for the definition and motivation of the term PSPACE-complete.)

Let $W = <Q,\Sigma,\Gamma,\delta,q_0,@,\$,F>$ be a linear bounded automata (LBA) where

Q is a (finite) set of states,
$\Sigma \subseteq \Gamma$ is the (finite) input alphabet,
Γ is the finite set of allowable tape symbols,
δ is the transition or next move function,
q_0 is the initial state,
$@,\$ \epsilon \Sigma$ are the left and right endmarker respectively, and
$F \subseteq Q$ is the set of accepting states.

Let $@a_1,...,a_l\$$ be an input to W. Without loss of generality, we assume that W accepts an input iff W halts for that input. Let "#" be a symbol not in Γ. Then one can construct a network (M,N) over the message set $Q \cup \Gamma \cup \{\#\}$, that will in some sense simulate the execution of W on $@a_1...a_l\$$. Furthermore, the network will have *each* of the following three properties:

(1) boundedness
(2) freedom from deadlocks
(3) freedom from unspecified receptions

iff W does not accept $@a_1...a_l\$$. In addition, M's input channel will be known to be bounded by l+3. Basically, the network will simulate W using the channels to contain the contents of the storage tape. M first sends the sequence of messages $@q_0\ a_1...a_n\$$ to N. Thus, the initial configuration of the LBA is represented in the channel. The machine N has only a single function. It can receive any message in $\Gamma \cup Q$ and then send the same message back to M. M then can simulate W by receiving a sequence of messages that represents a configuration of W and transmitting the sequence as it would be after the next move of W. That is, M receives and sends a sequence of message corresponding to the tape contents of W each time a move of W is simulated. If during the simulation an accepting state of W is entered then M can (nondeterministically) make additional moves that will result in the network becoming unbounded or reaching a deadlock or unspecified reception state. That is, M will, upon receiving a message $q \epsilon F$, have a choice of either entering a loop that sends only the message "#" to N or a loop that only receives messages. The aforementioned properties of the network are now clear, since N is not specified to receive the message "#".

Clearly then, the size of the network (M,N) will be polynomial in |Q|, $|\Gamma|$ and l. Consequently, if the algorithm given in Theorem 4 can be made to operate in polynomial time, then the acceptance problem for LBA's can be decided in polynomial time. But this implies that PSPACE=PTIME.

REFERENCES

1. Bochmann, G., Finite State Description of Communication Protocols, *Computer Networks*, Vol. 2, 1978, pp.361-371.

2. Brand, D. and Zafiropulo, P., On Communicating Finite-State Machines, *J. ACM*, Vol. 30, No. 2, April 1983, pp. 323-342.

3. Cook, S., Characterizations of Pushdown Machines in Terms of Time Bounded Computers, *J. ACM*, Vol. 18, 1971, pp. 4-18.

4. Cunha, P. and Maibaum, T., A Synchronization Calculus for Message-Oriented Programming, *Proc. 2nd International Conf. on Distributed Computing Systems*, April 1981, pp. 433-445.

5. Gouda, M., Manning, E. and Yu, Y., On the Progress of Communication between Two Finite State Machines, University of Texas, Department of Computer Sciences, Tech. Rep. No. 200, May 1982, revised August 1983.

6. Hopcroft, J. and Ullman, J., "Introduction to Automata Theory, Languages, and Computation", Addison-Wesley, Reading, Mass., 1979.

7. Jones, N., Lien, Y. and Laaser,W., New Problems Complete for Nondeterministic Logspace, *Math. Systems Theory*, Vol. 10, 1976, pp. 1-17.

8. Karp, R. and Miller, R., Parallel Program Schemata, *J. of Computer and System Sciences*, Vol. 3, No.2, 1969, pp.147-195.

9. Kosaraju, S., Decidability of Reachability in Vector Addition Systems, *Proc. of the 14th Annual ACM Symp. on the Theory of Computing*, 1982, pp. 267-281.

10. Mayr, E., An Algorithm for the General Petri Net Reachability Problem, *Proc. of the 13th Annual ACM Symp. on the Theory of Computing*, 1981, pp. 238-246.

11. Rubin, J. and West, C., An Improved Protocol Validation Technique, *Computer Networks*, Vol. 6, June 1982, pp. 65-73.

12. Sacerdote, G. and Tenney, R., The Decidability of the Reachability Problem for Vector Addition Systems, *Proc. of the 9th Annual ACM Symp. on Theory of Computing*, 1977, pp. 61-76.

13. Savitch, W., Relationships between Nondeterministic and Deterministic Tape Complexities, *J. of Computer and System Sciences*, Vol. 4, No. 2, 1970, pp. 177-192.

14. Sudborough, I., On Tape-Bounded Complexity Classes and Multihead Finite Automata, *J. Computer and Systems Sciences*, Vol. 10, No. 1, 1975, pp. 62-76.

15. Sunshine, C., Formal Modeling of Communication Protocols, USC/Inform. Sc. Institute, Research Report 81-89, March 1981.

16. Yu, Y. and Gouda, M., Deadlock Detection for a Class of Communicating Finite State Machines, *IEEE Trans. on Comm.*, Vol. COM-30, No. 12, December 1982, pp. 2514-2518.

17. Yu. Y. and Gouda, M., Unboundedness Detection for a Class of Communicating Finite State Machines, *Information Processing Letters*, 17 (1983), pp. 235-240.

18. Zafiropulo, P., et. al., Towards Analyzing and Synthesizing Protocols, *IEEE Trans. on Comm.*, Vol. COM-28, No. 4, April 1980, pp. 651-661.