PRIORITY NETWORKS OF COMMUNICATING FINITE STATE MACHINES

M.G.Gouda and L.E.Rosier

Department of Computer Sciences
University of Texas at Austin

TR-83-10 August 83

Abstract

Consider a network of two communicating finite state machines which exchange messages over two one-directional, unbounded channels, and assume that each machine receives the messages from its input channel based on some fixed (partial) priority relation. We address the problem of whether the communication of such a network is deadlock-free and bounded. We show that the problem is undecidable if the two machines exchange two types of messages. The problem is also undecidable if the two machines exchange three types of messages, and one of the channels is known to be bounded. However, if the two machines exchange two (or less) types of messages, and one channel is known to be bounded, then the problem becomes decidable. The problem is also decidable if one machine sends one type of message and the second machine sends two (or less) types of messages; the problem becomes undecidable if the second machine sends three types of messages. The problem is also decidable if the message priority relation is empty. We also address the problem of whether there is a message priority relation such that the priority network behaves like a FIFO network. We show that the problem is undecidable in general, and present some special cases for which the problem becomes decidable.

1. Introduction

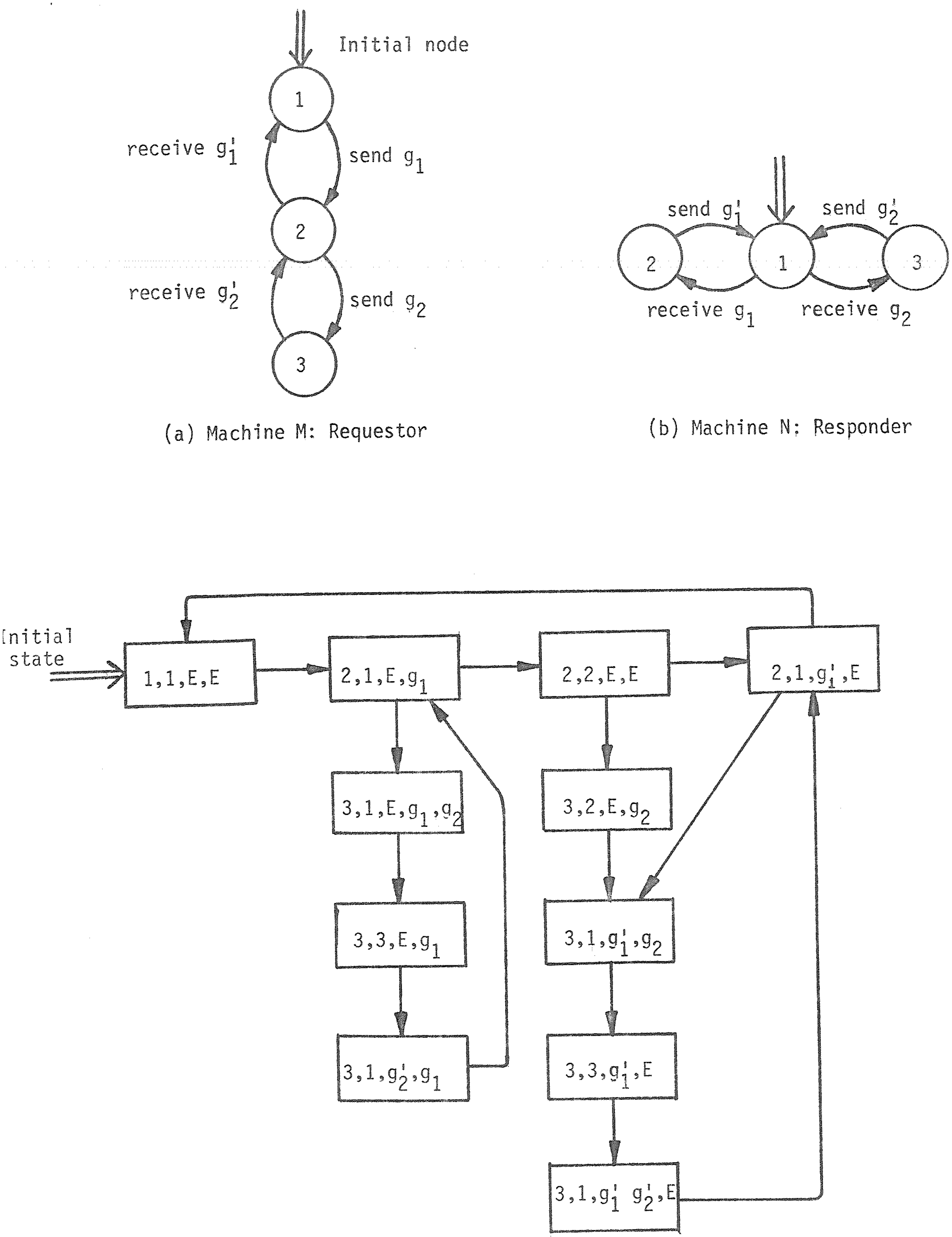
Networks of communicating finite state machines have proven extremely useful in the modeling [4], analysis [1,2,22], and synthesis [13,24] of communication protocols and distributed systems. However, most previous work (c.f. [1-4,13,17,22-24]) has focused on FIFO networks, i.e. networks where each machine receives the messages from its input channel based on the well known First-In-First-Out discipline. In this paper, we consider instead priority networks where messages are received based on a fixed, partial-ordered priority relation. There are two practical reasons to consider this class of networks:

i. In a number of existing communication protocols and distributed systems, messages are actually received based on a fixed priority relation rather than a FIFO discipline. (For example, INTERRUPT messages have a higher priority over sequenced messages in the packet layer of X.25 [20].) It is more appropriate to model and analyze such systems using priority networks than FIFO networks.

ii. In many cases, it is possible to select some fixed priority relation such that the resulting priority network behaves like a FIFO network. (For example in Section 7, we show that the Call Establishment and Clear procedures of the Binary Synchronous Protocol [11], which are usually modeled by a FIFO network, can be modeled by a priority network.)

In this paper, we consider a network of two communicating finite state machines that exchange messages over two unbounded, one-directional channels. 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.)

An example of a priority network is shown in Figure 1. It consists of two machines, machine M is called the requestor and machine N is called the responder. The requestor continuously sends a request message and receives a reply message. There are two types of requests, "regular" and "urgent"(denoted g_1 and g_2 respectively in Figure 1), and two types of replies, "regular" and "urgent" (denoted g'_1 and g'_2 respectively in Figure 1). After the requestor sends a g_1 message, it waits to receive a g'_1 message; however it can also send a g_2 message in which case it must first receive the corresponding g'_2 message before receiving the g'_1 message. This implies that g_2 and g'_2 have higher priorities than g_1 and g'_1 respectively. Figure 1c shows the "state reachability graph" of the network. Each node in this graph corresponds to one reachable state of the network, and is labelled by a four-tuple: The first (second) component refers to a node in machine M (N), and the third (fourth) component refers to the contents of the input channel of machine M(N), where E denotes the empty channel. Notice that the only next state after $[3,1,E,g_1g_2]$ is $[3,3,E,g_1]$, and not $[3,2,E,g_2]$; this is because g_2 has a higher priority than g_1.



(a) Machine M: Requestor

(b) Machine N: Responder

(c) Reachability graph

Figure 1 A priority network example

This model is equivalent, in computational power, to certain classes of extended Petri nets, in particular those with coloured or priority tokens [5,15]. However, the model presented here is more concise (since the channels and their contents are not modeled explicitly), and so is more convenient to use in modeling communication protocols and distributed systems.

Our results focus on the problem of whether the communication of a priority network is deadlock-free (i.e. the network can never reach a state after which no further progress is possible), and/or bounded (i.e. the number of reachable states is finite). We provide decidability/undecidablility results for these problems with respect to restricted classes of priority networks. We consider restrictions on the number of allowable message types and the size of the priority relation. We also examine the case where one of the channels is known to be bounded. The results presented here define sharp boundaries between the decidable and undecidable cases. They also depart considerably from similar results in the literature concerning FIFO networks[2,17].

The paper is organized as follows. In Section 2, the model of priority networks is presented formally. In Section 3, we show that the problem of detecting deadlocks and unboundedness is undecidable even if the machines exchange only two types of messages. We also consider the case where one of the two channels is known to be bounded, and show that three types of messages can make the problem undecidable in this case. (This problem is decidable in the case of FIFO networks[2].) Then in Section 4, we show that the same problem becomes decidable if only two types of messages are allowed. In Section 5, we consider the case of one of the two machines sending one type of message. We show that the problem is undecidable if the other machine sends three or more types of messages, and is decidable if the other machine sends two or less types of messages. (Both problems are decidable in the case of FIFO networks[17].) However, the latter result can be generalized to the case of three or more messages, if only two message types are mentioned in the priority relation. In Section 6, we examine the simplification (or reduction) of the message priority relation. In particular, we argue that if the message priority is reduced, and if the network after the reduction is deadlock-free and bounded, then the network before the reduction is also deadlock-free and bounded. Moreover, if the priority is reduced to the limit (i.e. all messages are received on a random basis), then the problems of detecting deadlocks and/or unboundedness are reduced to the reachability and unboundedness problems of vector addition systems [9,10,12,18], and so are decidable. In Section 7, we discuss how to select the message priority relation such that the priority network behaves like a FIFO network.

2. Priority Networks

A *message system* is an ordered pair $(G,\prec)$, where G is a finite, nonempty set of *messages*, and $\prec$ is a partial order over G called the *message priority relation*. If two distinct messages g_1 and g_2 in G are such that (g_1,g_2) is in $\prec$, denoted by $g_1 \prec g_2$, then g_2 is said to have a *higher priority* than g_1. The number $|G|$ of the messages in set G of a message system is called the *size* of the message system.

A *communicating machine* M over a message system $(G,\prec)$ is a finite 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 G. One of the nodes in M is identified as the *initial node*, and each node in M is reachable by a directed path from the initial node. For convenience, we assume that each node in M has at least one outgoing edge; outgoing edges of the same node have distinct labels. If the outgoing edges of a node are all sending (all receiving), then the node is called a *sending (receiving) node*; otherwise it is called a *mixed node*.

Let M and N be two communicating machines over the same message system $(G,\prec)$; the pair (M,N) is called a *priority network* of M and N.

A *state* of network (M,N) is a four-tuple [v,w,x,y], where v is a node in M, w is a node in N, and x and y are two multisets of messages in G. Informally, a state [v,w,x,y] of network (M,N) means that the execution of the two machines M and N has reached nodes v and w (respectively), while the input channels of M and N contain the multisets x and y (respectively) of messages.

The *initial state* of a priority network (M,N) is $[v_0,w_0,E,E]$, where v_0 is the initial node of M, w_0 is the initial node of N, and E is the empty multiset.

Let s=[v,w,x,y] be a state of a priority network (M,N), and let e be an outgoing edge of node v or w. A state s' of (M,N) 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' is obtained by adding exactly one g to y.

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

iii. e is a receiving edge, labelled receive(g), from v to v' in M, and x contains at least one g, and s'=[v',w,x',y] where x' is obtained by removing exactly one g from x, and if v has an outgoing edge labelled receive(g'), where $g\prec g'$, then x contains no g'.

iv. e is a receiving edge, labelled receive(g), from w to w' in N,and y contains at least one g, and s'=[v,w',x,y'] where y' is obtained by removing exactly one g from y, and if w has an outgoing edge labelled receive(g'), where $g\prec g'$, then y contains no g'.

The last parts of conditions iii and iv mean that messages are received in accordance with their priorities; the highest priority available message is received first. (Unrelated messages can be received in any order.)

Let s and s' be two states of a priority network (M,N); state s' is said to *follow* s iff there exists an edge e in M or N such that s' follows s over e.

Let s and s' be two states of network (M,N); state s' is *reachable from* s iff either $s=s'$ or there exist states $s_1, s_2, \ldots, s_r$ such that $s=s_1$, $s'=s_r$, and for $i=1,\ldots,r$, s_{i+1} follows s_i.

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

A reachable state $s=[v,w,x,y]$ of a priority network (M,N) is called a *deadlock state* iff the following three conditions are satisfied:

i. Both v and w are receiving nodes.

ii. Either $x=E$ (the empty multiset) or for any message g in x, there is no outgoing edge, from node v, labelled receive(g).

iii. Either $y=E$ or for any message g in y, there is no outgoing edge, from node w, labelled receive(g).

If no reachable state of network (M,N) is a deadlock state, then the communication of (M,N) is said to be *deadlock-free.*

Let (M,N) be a priority network over $(G,\prec)$. The input channel of machine M (N) is said to be *bounded* by some positive integer K iff for any reachable state $[v,w,x,y]$ of (M,N), $|x|$ ($|y|$) $\leq$ K, where $|x|$ is the number of messages in the multiset x. The communication of a network is *bounded* by K iff each of its two channels is bounded by K. If there is no such K, then the communication of (M,N) is *unbounded.*

Given a priority network of two communicating machines, it is often required to establish that its communication is both deadlock-free and bounded. Unfortunately, the problem of whether the communication of such a network is both deadlock-free and bounded is undecidable in general as is shown in the next section.

3. Undecidable Results

In the next two sections, we consider the problem of detecting deadlocks and unboundedness for three classes of priority networks:

1. The message system is of size less than or equal two.
2. The message system is of size less than or equal two, and one of the two channels is known to be bounded.
3. The message system is of size greater than or equal three, and one of the two channels is known to be bounded.

In this section, we show that the problem is undecidable for classes 1 and 3, and in Section 4, we show that the problem is decidable for class 2. (Notice that since the problem is undecidable for class 1 despite being decidable for class 2, the undecidability for class 1 must have risen from the added possibility of the two channels being unbounded.)

These results are interesting since they depart from the corresponding results for similar classes of FIFO networks. More specifically, the problem is decidable for classes 2 and 3 of FIFO networks, but remains undecidable for class 1 of FIFO networks [2].

Theorem 1: It is undecidable whether the communication of a class 1 priority network is both deadlock-free and bounded.

Proof: We show that any 2-counter machine T [14] (to be defined later), with no input, can be simulated by a priority network (M,N) over a message system (G,$\prec$) such that the following three conditions are satisfied:

i. $G=\{g_0,g_1\}$, and $\prec=\{g_0 \prec g_1\}$.

ii. The communication of (M,N) is deadlock-free.

iii. The communication of (M,N) is bounded by K iff the values of the two counters of T never exceed K.

Assume that there is an algorithm A to decide whether the communication of any such network is deadlock-free and bounded; then this algorithm can decide whether any 2-counter machine T halts as follows. First, construct from T a priority network which satisfies the above three conditions. Second, apply algorithm A to this network. If the answer is "yes", then (from conditions ii and iii above) T has a finite number of reachable configurations and its halting can be decided by exploring all the reachable configurations. If the answer is "no", then (from ii and iii above) T does not halt. Since it is undecidable whether any 2-counter machine halts [14], algorithm A cannot exist. It remains now to describe how to simulate any 2-counter machine by a priority network which satisfies the above three conditions; but first we define briefly 2-counter machines.

A 2-counter machine [6,14] is an offline deterministic Turing machine whose two storage tapes are semi-infinite, and whose tape alphabet contains only two symbols Z and B. The first (left-most) cells in both tapes are marked with Z symbols; all other tape cells are marked with B (for blank) symbols. Initially, each tape head scans the first cell of its tape. A nonnegative integer "i" can be stored at a tape by moving the tape head i cells to the right. Each move of the machine increments (decrements) , by one, each of the two stored numbers by moving the respective tape head one cell to the right (left). Each move of the machine depends on whether each of the two stored integers is currently greater than or equal zero. This is checked by examining the symbol in the currently scanned cell in each tape: A Z symbol indicates that the integer is zero, whereas a B symbol indicates that it is greater than zero.

Let T be a 2-counter machine; T can be simulated by a priority network (M,N) over (G,$\prec$), where $G=\{g_0,g_1\}$ and $\prec=\{g_0 \prec g_1\}$. Machine M simulates the finite control of T while N acts as an "echoer" that transmits the contents of its input channel to its output channel. At some instants, the number of g_1 messages in the network equals $2^{i}*3^{j}$ where i and j are the two integers currently stored in the counters of T. The g_0 messages are used for synchronization between M and N.

The simulation uses well known techniques from [14]. Each move of T is simulated by eight successive stages of moves executed by the two machines. Machine N executes stages 1,3,5, and 7, while machine M is waiting to receive a g_0 message. Machine M executes stages 2,4,6, and 8, while machine N is waiting to receive a g_0 message. Next, we describe these stages in more detail. (The four stages executed by N are identical; we describe only one of them namely stage 1.)

Stage 1: First M receives a g_0 message; it then receives all the g_1 messages from its input channel, and for each one, it sends one g_1 message. (N can determine that it has received the last g_1 by expecting to receive either g_0 or g_1. Since $g_0 \prec g_1$, then the last of the g_1 messages has already been received when N receives g_0. This same "trick" is used in all other stages.) Finally, N sends two g_0 messages, then awaits a g_0 message via its input channel which is currently empty. (Figure 2 shows machine N.)

Stage 2: First M receives a g_0 message; it then receives all the g_1 messages from its input channel, and for each one, it sends one g_1 message. After this M "determines" and "remembers" whether the number of received g_1 messages in this stage is divisible by 2. Finally, M sends two g_0 messages, then awaits a g_0 message. The next stage for M is stage 4.

Stage 4: Similar to stage 2 except that M determines and remembers whether the number of g_1 messages received in this stage is divisible by 3.

Stage 6: First, M receives a g_0 message. Now, M has determined the counter contents of T and is ready to simulate the T move. If T increments (decrements) the first counter, then M sends two (one) g_1 messages for each one (two) g_1 message it receives from its input channel. This has the effect of multiplying (dividing) by 2 the number of g_1 messages in the network. Finally, M sends two g_0 messages, then awaits a g_0 message.

Stage 8: Similar to stage 6 except that the number of g_1 messages in the network is multiplied or divided by 3 depending on the move of T being simulated.

The above simulation proceeds until T reaches a halting state in which case M starts to behave like N as an "echoer", i.e. M starts to transmit the contents of its input channel to its output channel executing stages similar to stage 1. It is straightforward to show that this network satisfies the above three conditions.[]

Theorem 2: It is undecidable whether the communication of a class 3 priority network is both deadlock-free and bounded. The result holds even if the message system is of size three and the message priority relation has two elements.

Proof: As in the proof of Theorem 1, we show that any 2-counter machine T can be simulated by a priority network (M,N) over $(G,\prec)$ such that the following three conditions are satisfied:

i. $G=\{g_0,g_1,g_2\}$, $\prec=\{g_0\prec g_1,\ g_0\prec g_2\}$, and the input channel of one machine, say M, is known to be bounded.

ii. The communication of (M,N) is deadlock-free.

iii. The communication of (M,N) is bounded by 2K+6 iff the values of the two counters of T never exceed K.

Machine M simulates the finite control of T while N acts as a "source" for the new messages to be added to the input channel of M. The number of g_1 (g_2) messages in the input channel of M corresponds to the integer stored in the first (second) counter of T. The g_0 messages are used for synchronization between M and N. Each move of T is simulated by the following five steps:

i. M waits to receive either g_0 or g_1. If it receives g_0, it recognizes that the value of the first counter is zero. If it receives g_1, it recognizes that the value of the first counter is greater than zero and waits until it receives g_0.

ii. Similar to step i except that message g_2 is waited for instead of g_1.

iii. M sends a g_0 message to N.

iv. On receiving g_0, N sends back six messages, two of type g_0, two of type g_1, and two of type g_2.

v. M receives zero or two messages of each of the two types g_1 and g_2 so that it increments or decrements the value of each counter according to the simulated move of T. (Figure 3 shows machine N.)

The above simulation proceeds until T reaches a halting state in which case M starts to preserve the contents of its input channel fixed. So in each move M sends one message g_0 to N, then receives all the six messages sent by N. It is straightforward to show that this network satisfies the above three conditions.[]

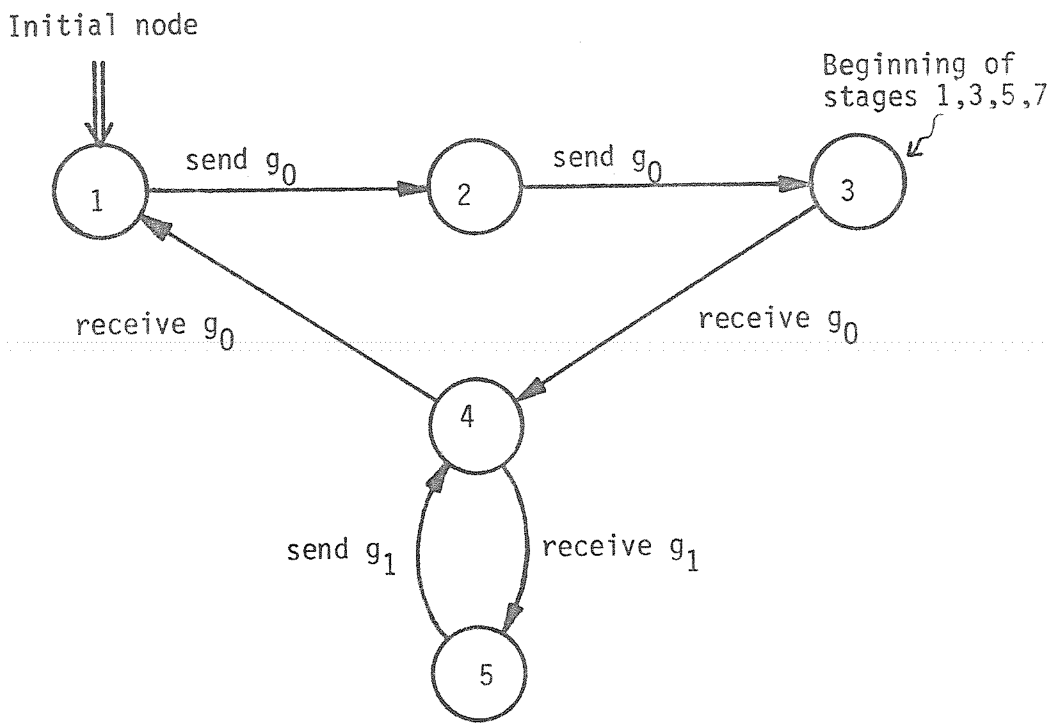


Figure 2 The "echoer" N in proof of Theorem 1

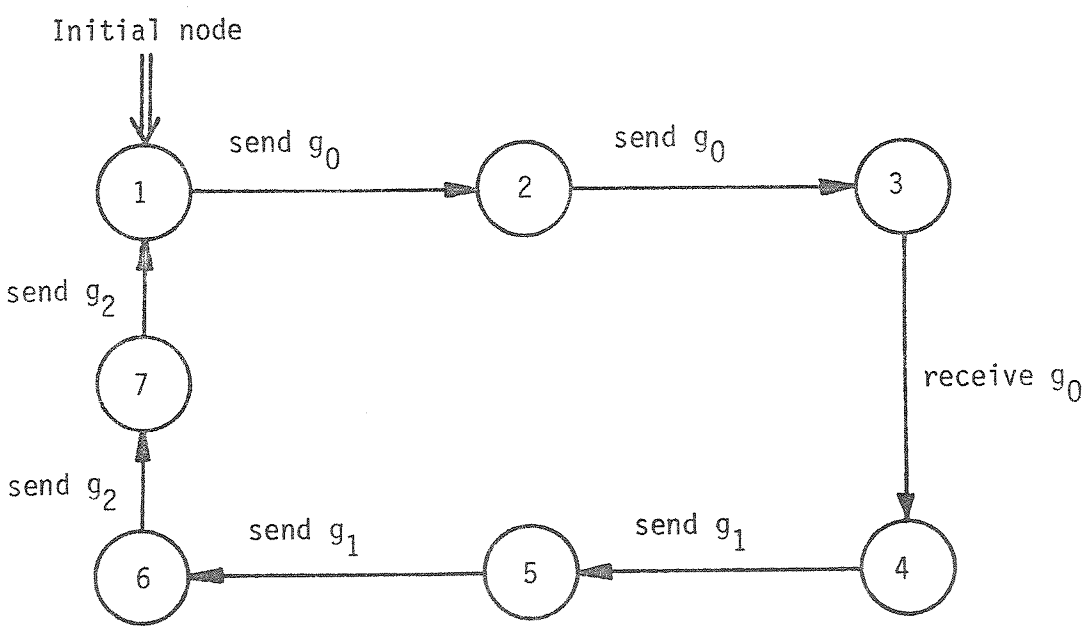


Figure 3 The "message source" N in proof of Theorem 2

The proofs of Theorems 1 and 2 show that the property of freedom of deadlocks and boundedness is undecidable. The same proofs also show that boundedness (by itself) is undecidable. To show that freedom of deadlocks (by itself) is undecidable, the proofs of Theorems 1 and 2 need to be modified slightly. The simulation of the 2-counter machine T proceeds as discussed before until T halts, in which case M enters a special node that has a self-loop labelled receive(g) for each message g in G. In other words, T halts iff (M,N) can reach a deadlock. This proves that freedom of deadlocks (by itself) is undecidable.

4. A Decidable Case: |G|=2 And One Channel Is Bounded

In this section, we show that detecting deadlocks and/or unboundedness for class 2 priority networks is decidable. For the sake of discussion in this section, let (M,N) be a class 2 priority network over $(G,\prec)$ where $G=\{g_1,g_2\}$ and $\prec=\{g_1\prec g_2\}$, and assume that the input channel of one machine, say M, is bounded by the positive integer K.

For i greater than or equal 0, define A(i) to be the set of all states [v,w,x,y] of (M,N), where multiset y contains at most i occurrences of message g_1, or i occurrences of message g_2. In what follows, we will be interested in the three sets A(0), A(L), and A(2L), where $L=(n+K+2)*\max(m,n)$, m=the number of nodes in machine M, and n=the number of nodes in machine N.

The possible contents of multiset y in each state [v,w,x,y] of (M,N) can be represented by "points" in the space illustrated in Figure 4. The points of the two g_1-g_2 axes (or Barrier 0) correspond to the states in A(0). The points between Barrier 1 and Barrier 0 correspond to the states in A(L). The points between Barrier 2 and Barrier 0 correspond to the states in A(2L).

Lemma 1: If network (M,N) reaches a state s not in A(L), then starting from s, M can stay "dormant" and N can progress (sending at most n messages) until (M,N) reaches a state in A(0).

Proof: Assume that (M,N) reaches a state s not in A(L). In this state, the input channel of N has at least (n+K+2)n messages of each type (g_1 and g_2). Then starting from s, M can stay dormant and N can progress until all occurrences of g_1 or all occurrences of g_2 are completely "depleted" from the input channel of N, i.e. the network (M,N) reaches a state s' in A(0). Notice that s' can be reached from s after at least (n+K+2)n steps (executed by N) since it takes at least that many steps to deplete all the messages of one type. It remains now to show that N can send at most n messages during the period from s to s'. This is shown by contradiction.

Assume that during the period from s to s', N sends n+1 messages, i.e. N traverses n+1 sending edges $e_1,\ldots,e_{n+1}$. Two of these edges say e_a and e_b, must be outgoing edges

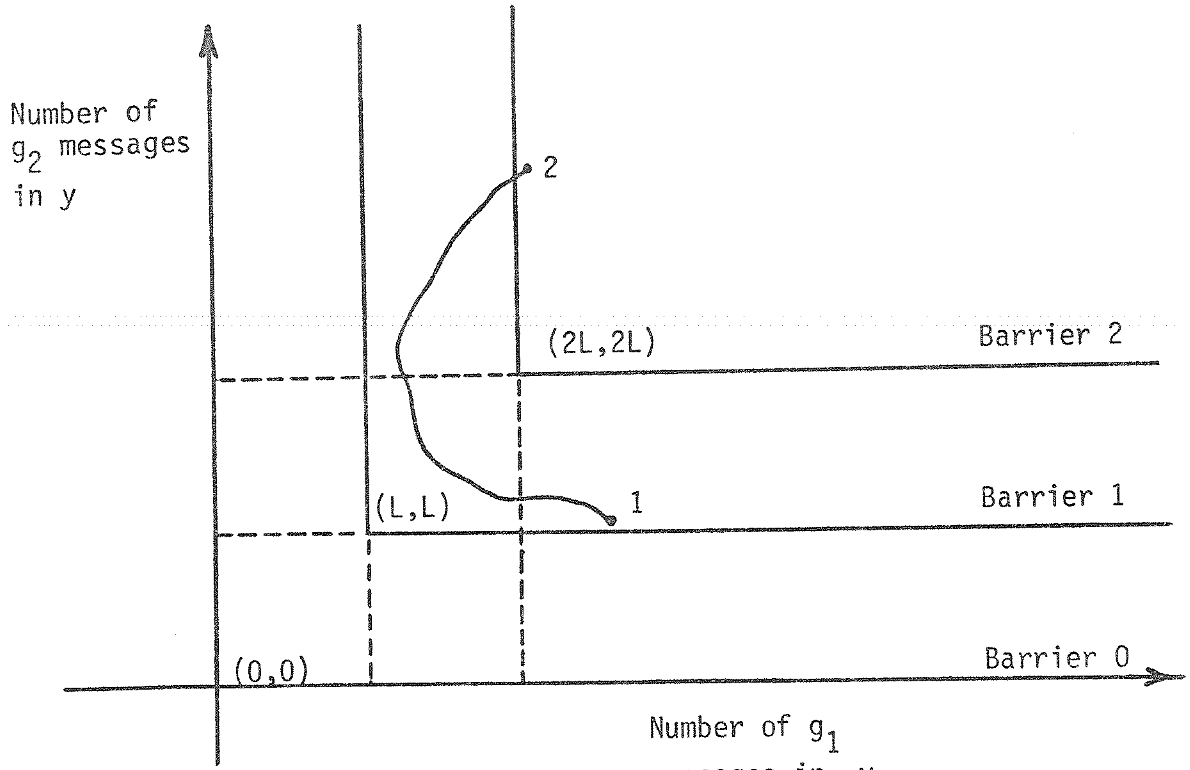


Figure 4 Possible contents of multiset y in each state (v,w,x,y) of (M,N)

of the same node d in N; node d must be in a directed cycle C (of length less than or equal n) that contains the sending edge e_a in N. Also, from the starting state s=[v,w,x,y], there must be a directed path P, from node w to node d, of length less than or equal n. Since N can execute at least (n+K+2)n steps starting from state s=[v,w,x,y], and since it can execute that many steps along any directed path starting from node w, then assume that N executes along path P then along cycle C for K+1 times. (This is possible since the length of this compound path is less than or equal (K+2)n which is less than (n+K+2)n.) However, along this compound path the sending edge e_a is traversed K+1 times causing N to send K+1 messages while M is dormant. This contradicts the assumption that the the input channel of M is bounded by K.[]

Lemma 2: If network (M,N) reaches a state s not in A(L) and later reaches a state s' such that none of the reached states from s to s' is in A(0), then N can send at most n messages during the period from s to s'.

Proof: (by contradiction) Assume that N sends n+1 messages during the the period from s to s', i.e. N traverses n+1 sending edges $e_1,...,e_{n+1}$ during this period. Since at state s the input channel of N has at least (n+K+2)n messages of each type, then N can traverse e_1 after at most n steps from state s. Also, N can traverse e_2 after at most n steps from traversing e_1, and can traverse e_3 after at most n steps from traversing e_2, and so on. In other words, N can traverse these n+1 sending edges after at most (n+1)n steps from s. Since (n+1)n is less than (n+K+2)n, then N can execute that many steps while M remains dormant. If N executes these (n+1)n steps while M remains dormant, the network will not reach a state in A(0) even though N has sent more than n messages; this contradicts Lemma 1.[]

Lemma 3: If network (M,N) reaches a state not in A(2L), then the communication of (M,N) is unbounded.

Proof: Assume that network (M,N) reaches a state not in A(2L), i.e. a state [v,w,x,y] where the contents of y are beyond Barrier 2. Let point 2, in Figure 3, correspond to the state of the network where Barrier 2 is first crossed. Also, let point 1, in Figure 3, correspond to the state of the network when it last crosses Barrier 1 up to the time of point 2. From Lemma 2, as the network progresses from point 1 to point 2, N can send at most n messages. Hence during this period, M can receive at most n+K messages and can send at least (n+K+2)m messages (to increase the contents of its input channel from L to 2L so that the network can reach point 2). Therefore, as the network progresses from point 1 to point 2, M must traverse more than m successive sending edges, i.e. it must traverse a cycle whose edges are all sending edges. Thus the communication of (M,N) is unbounded.[]

Theorem 3: It is decidable whether the communication of any class 2 priority network is both deadlock-free and bounded.

Proof: Let (M,N) be any class 2 priority network as defined at the beginning of this section. We show that this network can be simulated by a nondeterministic 1-counter machine T (to be defined) ,with no input, such that the communication of (M,N) is bounded iff there exists a constant C where the counter value of T never exeeds C. (A nondeterministic 1-counter machine is similar to the deterministic 2-counter machine mentioned in the previous section except that the machine's moves are allowed to be nondeterministic, and the machine has a single counter or tape instead of two[6,14].) Deciding whether there exists C such that the counter value of a 1-counter machine never exceeds C in every possible computation is Nondeterministic Logspace Complete (in the size of the machine) [17]. (Since these devices are essentially nondeterministic pushdown automata many decision problems are decidable[6]. Other related problems involving one counter automata (e.g. equivalence) have been studied. (See e.g. [7,8,21].)) Therefore boundedness of (M,N) can be decided, and so both boundedness and freedom of deadlocks can be decided. It remains now to show how to define T from (M,N).

The finite control of T is capable of remembering (at any instant):

i. the current nodes of M and N,

ii. the current contents of the (bounded by K) input channel of M (these can be represented by two integers between 0 and K, one integer for each message type), and

iii. the current contents of the (possibly unbounded) input channel of N up to 2L messages of each type (these can be represented by two integers between 0 and 2L, one integer for each message type).

T simulates the network (M,N) by choosing nondeterministically to simulate a move of M or N. Whenever the input channel of N has more than 2L messages of each type (indicating that the communication of (M,N) is unbounded by Lemma 3) T enters a state where it continuously increments its counter. Clearly the communication of (M,N) is bounded iff there exists C such that the counter value of T never exceeds C in every possible computation.[]

From the proof of Theorem 3, boundedness (by itself) is decidable for class 2 priority networks. The following theorem shows that freedom of deadlocks (by itself) is also decidable for the same class.

Theorem 4: It is decidable whether the communication of any class 2 priority network is deadlock-free.

Proof: Let (M,N) be any class 2 priority network as defined at the beginning of this section. We show that this network can be simulated by a nondeterministic 1-counter machine T such that the communication of (M,N) is deadlock-free iff T never halts. Deciding whether any nondeterministic 1-counter machine halts is Nondeterministic Logspace Complete (in the size of the machine) [17], and so deadlocks can be detected for class 2 priority networks.

The finite control of T is capable of remembering (at any instant):

i. the current nodes of M and N,

ii. the current contents of the (bounded by K) input channel of M,

iii. the current contents of the (possibly unbounded) input channel of N up to L messages of each type, and

iv. a string of length at most n over the messages g_1 and g_2.

T simulates the network (M,N) by choosing nondeterministically to simulate a move of M or N:

i. *Simulating a Move of M:* If the move can be simulated without causing more than L messages of each type to appear in the input channel of N, then the move is simulated directly. Otherwise, T must first simulate enough moves of N to bring down the number of one message type in this channel to L-1. However, some of these moves of N may send messages to M. T does not simulate these moves in the usual way; instead each sent message is concatenated to the right hand side of the string in the finite control of T. By Lemma 2, N cannot send more than n messages until its input channel is depleted of one type of message. At such a time the simulation can proceed directly again. After T has executed the above actions, the simulation of the original move of M can now take place.

ii. *Simulating a Move of N:* If the string in the finite control is empty, the move is simulated directly. Otherwise T moves the leftmost message of the stored string to the bounded channel. (That this may imply executing several moves of N is not important since only the message sent can affect the execution of M.)

Clearly T can reach a halting state iff (M,N) can reach a deadlock state.[]

In the proofs of Theorems 3 and 4, we have assumed that the bound of the bounded channel in any class 2 priority network is known. Suppose, however, that this is not the case. In this case, the bound can be obtained by an unbounded search (by Theorem 1 this is the best we can hope for) using the simulation procedure in the proof of Theorem 4. In this procedure, if the bounded channel is not bounded by some assumed value K, then the simulating 1-counter machine can reach a state where the bounded channel has K+1 messages.

5. The Case of One Machine Sending One Type of Message

Consider a priority network (M,N) over $(G,\prec)$, where one machine, say N, sends one type of message, i.e. there is a message g in G such that each sending edge in N is labelled send(g). The other machine M is assumed to send any number of message types from G. Let s_M denote the number of distinct message types sent by M. The decidability of the problem of whether the communication of any such network is both deadlock-free and bounded depends on the value of s_M:

i. If $s_M=1$, then the problem can be reduced [3,23] to the problem of "whether the reachability set of any vector addition system is finite?" which is decidable [10,12,18]. (See the next section.)

ii. If $s_M=2$, then the problem is decidable as discussed in this section.

iii. If s_M is greater than or equal 3, then the problem becomes undecidable. This can be shown using an identical proof to that of Theorem 2.

(These results for priority networks are different from the corresponding results for FIFO networks. The problem for FIFO network is always decidable, and in fact Nondeterministic Logspace Complete, regardless of the value of s_M [17].)

Theorem 5: Let (M,N) be a priority network over $(G,\prec)$, and assume that M sends two types of messages g_1 and g_2, where $g_1 \prec g_2$, and that N sends one type of message. The communication of (M,N) is unbounded iff one of the following two conditions is satisfied:

A. There are two reachable states $s=[v,w,x,y]$ and $s'=[v,w,x',y']$ such that the following three conditions hold:
  i. s' is reachable from s.
  ii. If state s' is reached from s via a state $s''=[v'',w'',x'',y'']$, then $|y''_2|>0$, where $|y''_2|$ is the number of g_2 messages in y''.
  iii. Either($|x|\leq|x'|$ and $|y_1|\leq|y'_1|$ and $|y_2|<|y'_2|$),
  or ($|x|\leq|x'|$ and $|y_1|<|y'_1|$ and $|y_2|\leq|y'_2|$,
  or ($|x|<|x'|$ and $|y_1|\leq|y'_1|$ and $|y_2|\leq|y'_2|$),
  where
  $|x|$ is the number of messages in x,
  $|y_i|$ (i=1,2) is the number of g_i messages in y, and
  $|y'_i|$ (i=1,2) is the number of g_i messages in y'.

B. There are two reachable states $s=[v,w,x,y]$ and $s'=[v,w,x',y']$ such that the following three conditions hold:
  i. s' is reachable from s.
  ii. $|y_2|=|y'_2|=0$.
  iii. Either($|x|\leq|x'|$ and $|y_1|<|y'_1|$),
  or ($|x|<|x'|$ and $|y_1|\leq|y'_1|$).

Proof:

If Part: We show that condition A is sufficient for the communication to be unbounded. (Proving that condition B is also sufficient for the communication to be unbounded is similar.) Assume that there are two reachable states s=[v,w,x,y] and s'=[v,w,x',y'] of (M,N) such that the three conditions in A hold. From i, state s' is reachable from s over a sequence of directed edges that form a directed cycle C_M (which starts and ends with node v) in M, and a directed cycle C_N (which starts and ends with node w) in N. From ii and iii, M and N can traverse the same two cycles any number of times. From iii, each time the two cycles are traversed, the number of messages in one channel increases while the number of messages in the other channel remains the same or increases. Therefore, the communication is unbounded.

Only If Part: We show that if the communication is unbounded and condition A is not satisfied, then condition B must be satisfied. Assume that the input channel of one machine, say M, is unbounded. Then, there is an infinite sequence of reachable distinct states $s_0,s_1,\ldots$ of (M,N) such that the following three conditions hold:

i. s_0 is the initial state of (M,N).

ii. For i=0,1,..., s_{i+1} follows s_i.

iii. For any K, there is a state s"=[v",w",x",y"] in the sequence such that |x"|>K.

Because this sequence is infinite, there must be a node pair (v,w), where node v is in M, node w is in N, and the pair (v,w) is repeated infinitely often in the states of the infinite sequence. Since this sequence does not satisfy condition A, it must have an infinite number of states s"=[v",w",x",y"] where the number of g_2 messages in y" equals zero. Because there are infinitely many of such states, condition B must be satisfied.[]

Based on the above theorem, the following algorithm can decide the boundedness of any priority network (M,N) over (G,≺), where N sends one type of message and M sends two types of messages g_1 and g_2, $g_1 \prec g_2$:

i. Let T be a directed rooted tree whose nodes are labelled with reachable states of (M,N) and whose directed edges correspond to the "follow" relation. Initially T has exactly one node labelled with the initial state of (M,N).

ii. **while** T has a leaf node n' labelled with a state s' that is followed by some state
do
if node n' has an ancestor node n labelled with state s in T such that s and s' satisfy condition A or B (in Theorem 5)
then stop: The communication of (M,N) is unbounded
else find all the states $s_1,\ldots,s_r$ which follow s';

add nodes $n_1,\ldots,n_r$ to T;
label each node n_i with state s_i;
add a directed edge from node n' to each n_i in T;

iii. **stop**: The communication of (M,N) is bounded.

Two comments concerning the above algorithm are in order:

i. The above algorithm is guaranteed to terminate. This is because (from the proof of Theorem 5) every infinite path whose nodes are labelled with distinct states in tree T must reach, after a finite number of nodes, two nodes whose state labels satisfy condition A or B.

ii. The above algorithm can decide boundedness; hence it can be used to decide both boundedness and freedom of deadlocks. This completes the proof for the following theorem.

Theorem 6: It is decidable whether the communication of a priority network, where one machine sends one type of message and the other machine sends two types of messages, is both deadlock-free and bounded.[]

Theorems 5 and 6 can be generalized in a straightforward fashion to the class of priority networks where one machine sends one type of message, the other machine sends an arbitrary number of message types, and the message priority relation is a singleton set. They can also be generalized to class 3 priority networks where the priority relation is a singleton set. Note that the class 3 priority network constructed in the proof of Theorem 2 is such that one machine sends one type of message and the message priority relation has two elements. Hence, this is the best that can be done.

6. Priority Reduction and the Decidability of the Random Reception Discipline

Let $(G,\prec_1)$ and $(G,\prec_2)$ be two message systems with the same set of messages. $(G,\prec_2)$ is called a *priority reduction* of $(G,\prec_1)$ iff $\prec_2$ is a subset of $\prec_1$, i.e. for any two messages g_1 and g_2 in G, if $g_1 \prec_2 g_2$, then $g_1 \prec_1 g_2$.

In Theorem 7 below, we show that if the priorities of a message system for some network is reduced, and if the resulting communication (after the priority reduction) is shown to be deadlock-free and bounded, then the original communication (before the reduction) is also deadlock-free and bounded. But first we prove the following lemma.

Lemma 4: Let R_1 denote a priority network (M,N) over $(G,\prec_1)$, and let R_2 denote the same network (M,N) over $(G,\prec_2)$. If $(G,\prec_2)$ is a priority reduction of $(G,\prec_1)$, then any reachable state of R_1 is a reachable state of R_2.

Proof: Assume that $(G,\prec_2)$ is a priority reduction of $(G,\prec_1)$; we show by induction that any reachable state of R_1 is a reachable state of R_2:

i. *Initial Step:* The initial state of R_1 is identical to that of R_2 since R_1 and R_2 have the same pair of communicating machines.

ii. *Induction Hypothesis:* Assume that s is a reachable state of both R_1 and R_2. We show next that any state s' which follows s in R_1 must also follow s in R_2.

iii. *Induction Step:* There are three cases to consider:

   a. If s' follows s by one machine (M or N) sending a message in R_1, then s' follows s by the same machine sending the same message in R_2.

   b. If s' follows s by one machine receiving a message g in R_1, and if every ordered pair involving message g in $\prec_1$ is also in $\prec_2$, then s' follows s by the same machine receiving the g in R_2.

   c. Assume that s' follows s by one machine receiving message g in R_1, and that some ordered pairs involving g are in $\prec_1$ but not in $\prec_2$. Then also in this case, s' follows s by the same machine receiving g in R_2.[]

The next theorem follows immediately from the above lemma.

Theorem 7: Let R_1 and R_2 be as defined in Lemma 1, and assume that $(G,\prec_2)$ is a priority reduction of $(G,\prec_1)$. If the communication of R_2 is deadlock-free and bounded, then the communication of R_1 is deadlock-free and bounded.[]

Theorem 7 is useful iff priority reduction can lead to simpler proofs for freedom of deadlocks and boundedness. From the discussion at the end of Section 5, if the priority relation is reduced to a single element, then the problem becomes decidable for priority networks where one machine sends only one type of message and for class 3 priority networks. Also, the next theorem states that if the priorities are reduced to the limit (i.e. all sent messages are of equal priorities, and so are received on a random basis), then the problem of whether the communication is both deadlock-free and bounded becomes decidable.

Theorem 8: (The Random Reception Theorem) It is decidable whether the the communication of any priority network, with empty message priority relation, is deadlock-free and/or bounded.

Sketch of the Proof: Any priority network (M,N) over $(G,\prec)$, where $\prec$ is empty can be simulated [15] by a vector addition system U such that the communication of (M,N) is bounded iff the reachability set of U is finite. Finiteness of the reachability sets of vector addition systems is decidable [9], and so is boundedness for priority networks with empty message priority relations. Therefore, the property of both freedom of deadlocks and boundedness is also decidable.

Also, any priority network with empty message priority relation can be simulated by a vector addition system such that the network can reach a deadlock state iff the reachability set of the vector addition system contains a predefined finite set of vectors [15]. The reachability problem of vector addition systems is decidable [10,12,18], and so is freedom of deadlocks (by itself) for priority networks with empty message priority relations.[]

It is straightforward to show that Theorem 8 can be generalized to the case of a priority network with r communicating machines (r greater than or equal 2) provided that the message priority relation is empty.

7. Achieving the FIFO Discipline Using Priorities

From Theorem 1 (or 2), priority networks can simulate any 2-counter machine; therefore they can simulate any FIFO network. In this section, we discuss the following special type of simulation. Given two communicating machines M and N whose message labels are taken from a finite set G of messages, is there a message priority relation $\prec$ such that the priority network (M,N) over $(G,\prec)$ "behaves like a FIFO network". But before we define how a priority network behaves like a FIFO network, we first need to add more structure to the concept of a state of a priority network.

As defined in Section 2, a state of a priority network is a four-tuple [v,w,x,y], where both x and y are multisets of messages. We adopt the following convention:

i. Both x and y are represented as strings of messages.

ii. When a machine M (N) sends a message g, then g is concatenated to the right hand side of y (x) yielding y.g (x.g), where "." is the string concatenation operator.

iii. When a machine M (N) receives a message g, then the left-most occurrence of g in x (y) is removed.

From i and ii, if a message g is to the left of a message g' in x or y, then g must have been sent "before" g'. This implies that the left-most message in x (y) is the current "oldest" message in x (y). From iii, whenever a machine M or N receives a message g, it must receive the oldest available copy of this message. Notice that this convention does not violate the state reachability of a priority network; it merely indicates for any reachable state [v,w,x,y], the order in which the messages in x and y have been sent.

A priority network (M,N) over $(G,\prec)$ is said to *behave like a FIFO network* iff the following four conditions are satisfied for any reachable state [v,w,x,y] of the network:

i. If x=g.x' and v has an outgoing edge labelled receive(g), then v has no outgoing edge labelled receive(g') for any other message g' in x'.

ii. If y=g.y' and w has no outgoing edge labelled receive(g), then w has no outgoing edge labeled receive(g') for any other message g' in y'.

iii. If x=g.x' and v has an outgoing edge labelled receive(g), then $g' \prec g$ for any other message g' in x' where v has an outgoing edge labelled receive(g').

iv. If y=g.y' and w has an outgoing edge labelled receive(g), then $g' \prec g$ for any other message g' in y' where w has an outgoing edge labelled receive(g').

The question "Given M,N, and G, is there a $\prec$ such that (M,N) over $(G,\prec)$ behaves like a FIFO network?" may have a positive or negative answer depending on the given M,N, and G. For example, the answer for the two machines in Figure 5 is "no", and the answer for the two machines in Figure 6 is "yes". (If the priority network in Figure 6 behaves like a FIFO network, then it models the call establishment and clear procedures for the Binary Synchronous Protocol [11], where

machine M models the primary station,
machine N models the secondary station,
message g_1 is the "initial inquiry" message SYN SYN ENQ,
message g_2 is the "try again" message SYN SYN WACK,
message g_3 is the "negative ACK" message SYN SYN NACK,
message g_4 is the "positive ACK" message SYN SYN ACK, and
message g_5 is the "clear" message SYN SYN EOT.)

Unfortunately, the above question is undecidable in general. A proof of this can be outlined as follows. Simulate any 2-counter machine T using a priority network (M,N) over $(G,\prec)$ that behaves like a FIFO network until T reaches a halting state in which case M and N start to execute the two machines in Figure 5 (i.e. those whose priority network does not, for any $\prec$, behave like a FIFO network). Thus T halts iff there is no $\prec$ such that (M,N) over $(G,\prec)$ behaves like a FIFO network. It remains now to describe a priority network which simulates T while behaving like a FIFO network. One such network is as follows. One machine M simulates the finite control of T while the other machine N acts as an echoer which sends back each message it receives from M. The contents of the two counters of T are usually stored in the in input channel of M as: $g_1g_1 \cdots g_1g_0g_2g_2 \cdots g_2g_0$, where the number of occurrences of g_1 (g_2) represents the contents of the first (second) counter, and each occurrence of g_0 acts as a separator between the g_1 messages and the g_2 messages. For M to simulate one move of T, it must make two complete passes on the contents of the input channel:

i. In the first pass, M receives all the messages from its input channel, one by one, and sends them without change to its output channel. (The objective of this pass is for M to decide whether the value of each counter is zero or greater than zero.) Since the echoer N returns these messages to the input

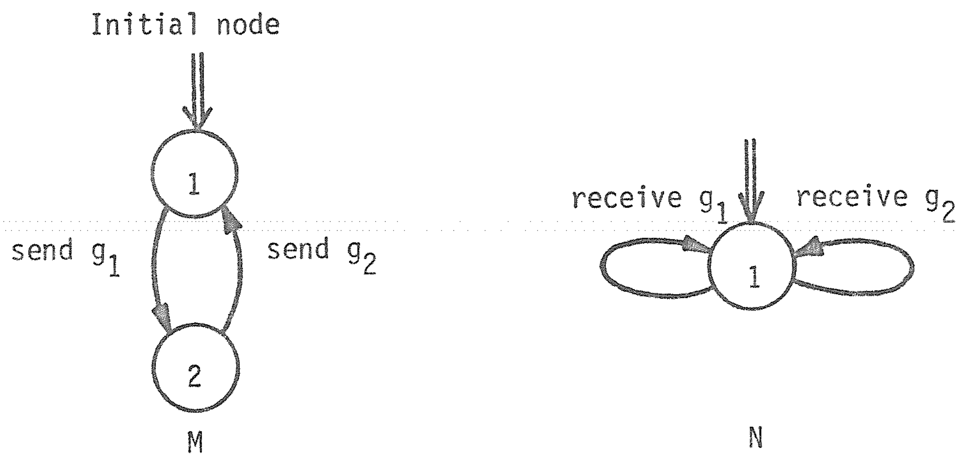
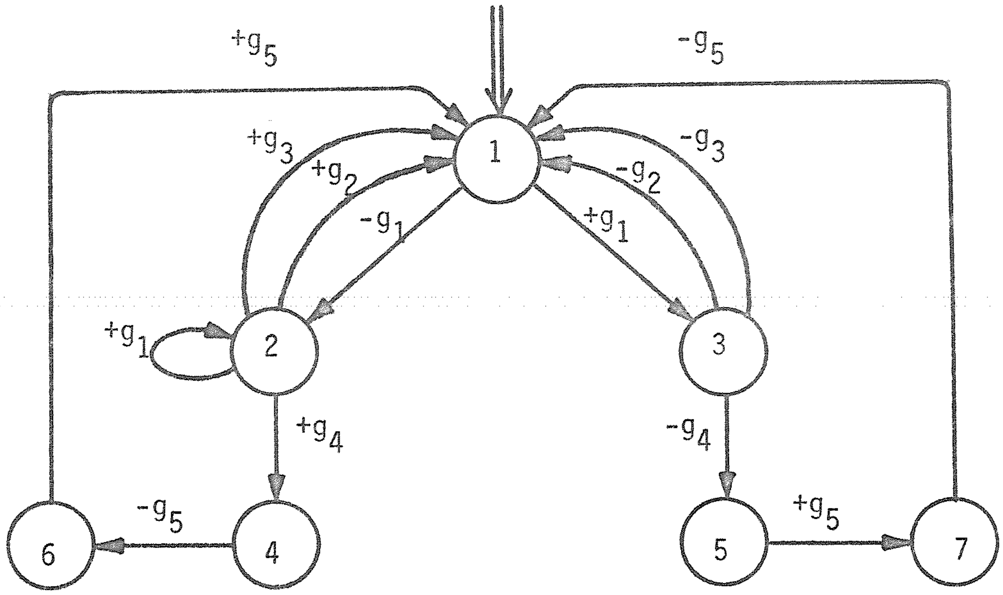
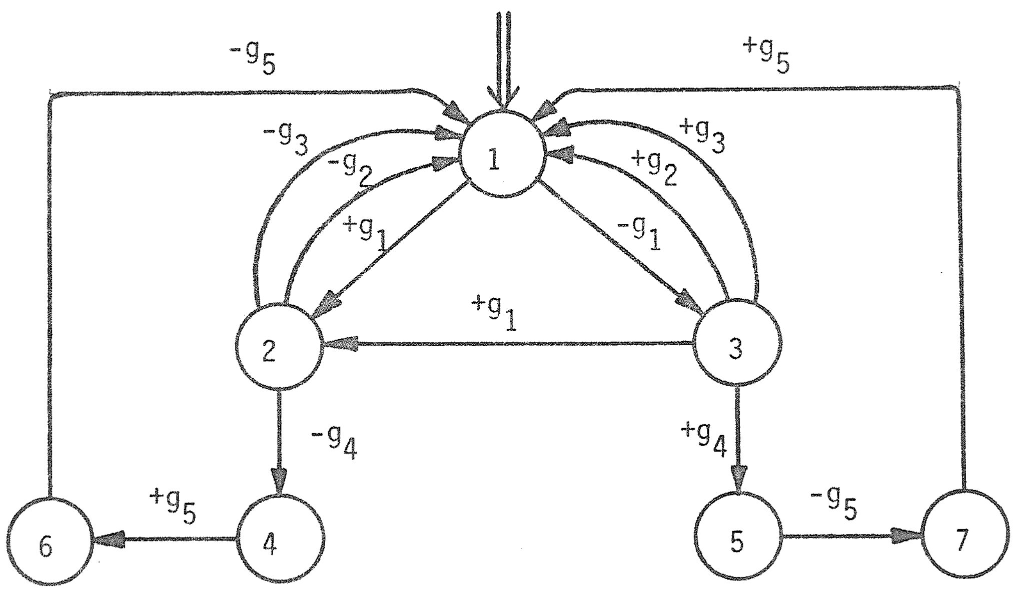


Figure 5 Two communicating machines whose priority network cannot behave like a FIFO network



M



N

Figure 6 Two communicating machines whose priority network with message priority

$\prec = \{ g_1 \prec g_2,\ g_1 \prec g_3,\ g_1 \prec g_4 \}$

behaves like a FIFO network (Notation: -g means send g, +g means receive g) channel of M, and in order to avoid mixing the returned messages with the original messages, N must change each g_1 message to a g_3 message and each g_2 message to a g_4 message; the g_0 messages remain the same.

ii. In the second pass, M receives all the messages from its input channel, one by one, and sends them, after performing the appropriate action of T, to its output channel. N returns the messages to the input channel of M after changing each g_3 to g_1, and each g_4 to g_2; the g_0 messages remain the same.

It is straightforward to show that each receiving node in M or N has exactly two outgoing edges and that one of these edges is labelled receive(g_0). Therefore by selecting the message priority relation $\prec=\{ g_0\prec g_1, g_0\prec g_2, g_0\prec g_3, g_0\prec g_4 \}$, the priority network (M,N) over $(G,\prec)$ behaves like a FIFO nework. This completes the proof of the following theorem.

Theorem 9: It is undecidable whether, for any two communicating machines M and N whose message labels are taken from a set G, there exists a message priority relation $\prec$ such that (M,N) over $(G,\prec)$ behaves like a FIFO network.[]

There are special cases for which the above problem becomes decidable. For instance, if the communication between M and N, assuming a FIFO discipline, is bounded (i.e. the number of distinct reachable states is finite), then the problem can be decided by the following straightforward algorithm:

i. Initially, $\prec$ is the empty set.

ii. **for** each reachable state [v,w,x,y] of (M,N) assuming a FIFO discipline
do
if x (y) equals g...g'..., and v (w) has an outgoing edge labelled receive(g') and
no outgoing edge labelled receive(g)
then stop: no $\prec$' can make the (M,N) over $(G,\prec')$ behave like a FIFO network

elsif x (y) equals g.z and v (w) has an outgoing edge labelled receive(g)
then for any other g' in z where v (w) has an outgoing edge labelled receive(g'),
add the element $g'\prec g$ to the set $\prec$

iii. **if** the resulting $\prec$ is a partial order
then stop: (M,N) over $(G,\prec)$ behaves like a FIFO network
else stop: no $\prec$' can make (M,N) over $(G,\prec')$ behave like a FIFO network

This algorithm can be applied to machines M and N in Figure 6, whose communication, assuming a FIFO discipline, is bounded. The result is the message priority relation $\prec=\{ g_1\prec g_2, g_1\prec g_3, g_1\prec g_4 \}$ that makes the priority network behave like a FIFO network.

The above decidability algorithm operates on the reachable state space of the network; hence it yields exponential complexity. In some other cases the decidability algorithm needs only to operate on the directed graphs of the two machines yielding polynomial complexity. One such a case is where the two communicating machines are "compatible" as defined next.

Two communicating machines M and N are called *compatible* iff the directed graphs of M and N are isomorphic as follows:

i. For every sending (receiving) node in one machine, there is a receiving (sending) node in the other machine.

ii. Neither machine has any mixed nodes.

iii. For every sending (receiving) edge labelled send(g) (receive(g)) in one machine, there is a receiving (sending) edge labelled receive(g) (send(g)) in the other machine.

If a priority network (M,N) of two compatible machines behaves like a FIFO network, then its communication is guaranteed to be deadlock-free [4]. Moreover, if each directed cycle in M or N has at least one sending and one receiving edge, then the communication is also bounded [4]. The next theorem states that it is decidable whether a priority network of compatible machines behaves like a FIFO network. The decidability algorithm (in the theorem's proof) operates on the directed graphs of the two machines yielding polynomial complexity.

Theorem 10: It is decidable whether for any two compatible machines M and N whose message labels are taken from a set G, there exists a message priority relation $\prec$ such that (M,N) over $(G,\prec)$ behaves like a FIFO network.

Proof: We first present a decidability algorithm for the problem, then prove its correctness. The algorithm consists of three steps:

i. Initially, $\prec$ is the empty set.

ii. **for** each receiving node u in M or N, and
    for each two distinct messages g and g' in G
  do
  if there are two outgoing edges labelled receive(g) and receive(g') from u, and if
    there is a directed path of all receiving edges from the edge labelled receive(g) to
    an edge labelled receive(g')
  then add $g' \prec g$ to the set $\prec$

iii. **if** the resulting $\prec$ is a partial order
  then stop: (M,N) over $(G,\prec)$ behaves like a FIFO network

else stop: no $\prec$' can make (M,N) over (G,$\prec$') behave like a FIFO network

If Part: Assume that the resulting $\prec$ of step ii is a partial order; we show that (M,N) over (G,$\prec$) behaves like a FIFO network. Let s=[v,w,x,y] be the first reachable state, from the initial state, that does not satisfy the definition of "behave like a FIFO network". Without loss of generality, assume that the problem is with the v and x components of s. Therefore, state s must satisfy at least one of the following two conditions:

i. x=g.x' where x' contains a message g' distinct from g, and v has an outgoing edge labelled receive(g'), and has no outgoing edge labelled receive(g).

ii. x=g.x' where x' contains a message g' distinct from g, and v has two outgoing edges labelled receive(g) and receive(g'), and g'$\prec$g is not in $\prec$.

Since s is the first reachable state that violates the FIFO behaviour and since the two machines M and N are compatible, M and N must have reached s via two directed paths p and q such that $p=\langle a,b,\ldots,v\rangle$ and $q=\langle a',b',\ldots,v',\ldots,w\rangle$, where the nodes a',b',...,and v' in N correspond to the nodes a,b,...and v in M (respectively). Moreover, the directed path $\langle v',\ldots,w\rangle$ must consist entirely from the sending edges which have sent the message sequence x=g.x'. Thus v' must have an outgoing edge labelled receive(g). Because of the compatibility of M and N, v must have an outgoing edge labelled receive(g). Therefore, condition i cannot be satisfied. Also message g' must have been sent along the the path $\langle v',\ldots,w\rangle$ in N; it must be expected along the corresponding path of all receiving edges in M. Hence g'$\prec$g must have been added to $\prec$ in step ii of the decidability algorithm, and condition ii cannot be be satisfied.

Only If Part: Assume that the resulting $\prec$ is not a partial order. Assume also that there is a partial order $\prec$' such that (M,N) over (G,$\prec$') behaves like a FIFO network. Since $\prec$' is a partial order while $\prec$ is not, there must be an element g'$\prec$g in $\prec$ but not in $\prec$'. In other words, there must be a node v in M or N with two outgoing edges labelled receive(g) and receive(g'), and there must be a directed path of all receiving edges from the edge labelled receive(g) to an edge labelled receive(g'). Without loss of generality, assume that this node v is in M. Because (M,N) over (G,$\prec$') behaves like a FIFO network, and because M and N are compatible, it is possible that the network reaches a state s=[v,v',E,E], where v' is the sending node (in N) that corresponds to the receiving node v in M, and E denotes the empty string. From state s, N can send a sequence x of messages starting with g and ending with g' (x=g...g') guiding the network into a state s'=[v,w,x,E]. This reachable state s' violates condition ii which is required for the network to behave like a FIFO network. Therefore, there is no partial order $\prec$' such that (M,N) over (G,$\prec$') behaves like a FIFO network. []

8. 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. 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.

4. 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.

5. Hack, M., Decidability Questions for Petri Nets, Ph.D. dissertation, Department of Electrical Engineering, MIT, 1975.

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

7. Ibarra, O., Reversal-Bounded Multicounter Machines and their Decision Problems, *J. ACM*, Vol. 25, No. 1, 1978, pp. 116-133.

8. Ibarra, O. and Rosier, L., On Restricted One-Counter Machines, *Math. Systems Theory*, Vol. 14, 1981, pp. 241-245.

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

10. 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.

11. Lam, S., Data Link Control Procedures, in "Computer Communications, Volume I Principles"(W. Chou, ed.), Prentice Hall, Englewood Cliffs, NJ, 1983, pp. 81-113.

12. 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.

13. Merlin, P. and Bochmann, G., On the Construction of Communication Protocols and Module Specification, Pub. 352, Dept. dinformatique de recherche op' erationnelle, Universit'e de Montreal, January 1980.

14. Minsky, M., "Computation: Finite and Infinite Machines", Prentice Hall, Englewood Cliffs, NJ, 1967.

15. Peterson, J., "Petri Net Theory and the Modeling of Systems", Prentice Hall,

Englewood Cliffs, NJ, 1981.

16. Rackoff, C., The Covering and Boundedness Problems for Vector Addition Systems, *Theor. Comput. Sci.*, Vol. 6, 1978, pp. 223-231.

17. Rosier, L. and Gouda, M., On Deciding Progress for a Class of Communication Protocols, in preparation, August 1983.

18. 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.

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

20. Tannenbaum, A., "Computer Networks", Prentice Hall, Englewood Cliffs, NJ, 1981.

21. Valiant, L. and Paterson, M., Deterministic One-Counter Automata, *J. of Computer and System Sciences*, Vol. 10, 1975, pp. 340-350.

22. 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.

23. Yu. Y. and Gouda, M., Unboundedness Detection for a Class of Communicating Finite State Machines, to appear in *Information Processing Letters*.

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