PROVING LIVENESS FOR NETWORKS OF COMMUNICATING FINITE STATE MACHINES

M. G. Gouda and C. K. Chang

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

TR-84-04 February 1984

ABSTRACT

Consider a network of communicating finite state machines that exchange messages over unbounded, FIFO channels. In general, for a node in one of the machines to be live (i.e. be executed by its machine infinitely often during the course of communication), each machine in the network should progress in some fair fashion. We define three graduated notions of fair progress (namely weakly fair, fair, and strongly fair), and based on them we define three corresponding degrees of node liveness (namely strongly live, live, and weakly live, respectively). We discuss techniques to verify that a given node is weakly live, live, or strongly live in its network. These techniques can be automated, and they are effective even if the network under consideration is unbounded (i.e. has an infinite number of reachable states). We use our techniques to establish the liveness of some real communication protocols; these include an unbounded Start-stop protocol, an unbounded Alternating Bit protocol, and a simplified version of the CSMA/CD protocol for local area networks.

Table of Contents

1. Introduction

Consider a network of some finite state machines that communicate exclusively by exchanging messages via connecting channels. There are two one-directional, unbounded, FIFO channels between any two machines in the network. Each machine has a finite number of states and state transitions, and each state transition is accompanied by either sending a message to one of the machine's output channels, or receiving a message from one of the machine's input channels.

Networks of communicating finite state machines are useful in modeling [5,25], analysis [2,3,10,11,12,13,16], and synthesis [4,7,8,14,18,31] of communication protocols, and distributed systems. The analysis problem for these networks can be stated as follows: "Given an arbitrary network of communicating finite state machines, prove that the communications within the network will satisfy some desirable properties." Most of the work to solve this problem has concentrated so far on properties such as boundedness [30], freedom of deadlocks [26,27,28,29] and unspecified receptions [10]. These are all safety properties [22]; i.e. they merely guarantee that nothing bad will happen during the course of communication. In order to guarantee that something good will happen (infinitely often), we need to establish some liveness properties for the given network. In this paper, we identify some liveness properties for networks of communicating finite state machines, and present techniques to prove these properties for such networks.

The pioneering works of Pnueli [23,24], Owicki and Lamport [22], Misra and Chandy [20,21], and Hailpern and Owicki [15] have established the foundations for defining and proving general liveness properties of concurrent programs and distributed systems. Pnueli [23,24] has introduced a version of temporal logic as a tool to specify and verify properties of concurrent programs. Owicki and Lamport [22] have used Pnueli's temporal logic to define "proof lattices", that are both rigorous and easy to understand, to verify liveness properties of concurrent programs. Later, Hailpern and Owicki [15] have used the same temporal logic in a modular verification methodology for distributed systems, where the intended system's assertions are verified using the assertions of the different processes in the system.

Misra and Chandy [20,21] have introduced a novel and modular verification methodology for distributed systems, that is not based on temporal logic. In their methodology, safety and liveness properties of any process in the system are defined by three assertions. Applying a "theorem of hierarchy", the process' assertions can be used to verify the safety and liveness properties of the whole system (which are also defined by three assertions).

The above approaches outline general frameworks where many types of liveness properties can be defined and verified for general systems. In this paper, we restrict our attention to defining and proving liveness for networks of communicating finite state machines. This is a significant restriction. For instance, under this restriction, there is no need for a general purpose temporal logic to define the required liveness properties of

these networks. In fact, we feel that only one property needs to be defined and verified for any such network, namely that some state(s) in some machine(s) in the network will be reached infinitely often during the course of communication under some fairness assumption. (We characterize three graduated fairness assumptions that lead to three degrees of liveness.) These restrictions and simplifications have led to sufficient conditions that can be checked algorithmically for any given network, and if established for a network, can ensure the liveness of that network.

The paper is organized as follows: Networks of communicating finite state machines are defined in Section 2. The notions of "infinitely often" and "fairness" are defined in terms of three classes of "fair communication sequences" in Section 3. In Section 4, we use these fair communication sequences to define three degrees of liveness for networks of communicating finite state machines. Also in this section, we discuss some simple sufficient conditions to establish liveness. Stronger sufficient conditions to establish liveness are discussed in Section 5. Then, in Section 6, we apply our techniques to establish the liveness of three networks that model some practical protocols. Concluding remarks are in Section 7.

All the results in this paper are applicable to networks with any number of communicating machines. But for the sake of clarity, we carry most of the discussion on networks with only two machines. Later in Section 6, we discuss a network example with three machines to illustrate how to apply our results in this case.

2. Networks of 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 (or receiving) edge is labelled -g (or +g, respectively) for some *message* g in a finite set G of messages. One of the nodes in M is identified as its *initial node*, and each node in M is reachable by a directed path from the initial node. For convenience, each node in M has at least one outgoing edge, and the outgoing edges of the same node must have distinct labels. A node in M whose outgoing edges are all sending (or all receiving) edges is called a *sending* (or *receiving*, respectively) *node*; otherwise it is called a *mixed node*.

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] means that the executions of M and N have reached nodes v and w respectively, while the input channels of M and N have the message strings 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 is satisfied:

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

ii. e is a sending edge, labelled -g, form w to w' in N, and s'=[v,w',x.g,y].

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

iv. e is a receiving edge, labelled +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 s and s' be two states of (M,N), s' is *reachable from* s 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.

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

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

In this paper, we characterize the notion of liveness for networks of communicating finite state machines. In particular, we argue that the liveness of such a network should guarantee that "something good" will "occur infinitely often" during the course of communication. This is better explained by an example.

Example 1 : A sender and a receiver exchange messages over a communication medium that may corrupt transmitted messages. The sender sends data messages to the receiver. For each received message, the receiver responds by sending back a positive acknowledgement (if the received message has not been corrupted) or a negative acknowledgement (if the received message has been corrupted by the medium). When the sender receives a positive acknowledgement, it sends the next data message, and when it receives a negative acknowledgement or a corrupted message, it resends the last data message. (It is assumed that data messages have sequential numbers that enable the receiver to detect message duplications.)

This protocol is modeled by the two communicating machines M and N in Figure 1a, where M models the sender, N models the receiver, and the exchanged messages are

as follows:

Ndata	denotes a next data message.
Ldata	denotes a last data message.
Ack	denotes a positive acknowledgement message.
Nack	denotes a negative acknowledgement message.
Err	denotes a corrupted message.

Notice that the medium is not modeled explicitly as a separate machine. Instead, its effect is simulated by both M and N. So at any node in M or N with an outgoing sending edge where a message g is sent, we added another outgoing sending edge, where a corrupted message Err is sent. Traversing this added edge simulates the effect of message g being sent then later corrupted by the medium into an erroneous message.

The liveness requirement of this network can be stated by requiring that node 1 in machine M be "reached infinitely often" during the course of communication between M and N. (Notice that if node 1 in M is reached infinitely often, then the sender M must have sent Ndata messages to the receiver N infinitely often.) In general, it is possible that M and N communicate forever without node 1 being reached infinitely often. For instance, if M persists on sending Err messages, then node 1 will never be reached again. Therefore, to ensure that node 1 in M will be reached infinitely often, it is important to assume that both M and N will behave in some fair fashion during their communication. (This fairness assumption is equivalent to the assumption that the corrupting communication medium between the sender and receiver is "fair". In other words, if the medium receives an infinite number of messages, then an infinite number of them will be delivered without corruption and an infinite number of them will be corrupted.) The notion of fairness for networks of communicating finite state machines is discussed next.

□

3. Fair Communication Sequences

A *communication sequence* of a network (M,N) is a, possibly infinite, sequence $<s_0,s_1,...>$ of reachable states of (M,N) such that s_0 is the initial state of (M,N), and s_{i+1} follows s_i, i=0,1,... . If a communication sequence has a state that cannot be followed by any other state, then the sequence is finite, otherwise it is infinite.

A network (M,N) is called *safe* iff the following two conditions hold:

i. Each communication sequence of (M,N) is infinite.

ii. If any communication sequence $<s_0,s_1,...>$ of (M,N) has a state s_i=[v,w,x,y] such that v(w) is a receiving node and x(y) = E (the empty string), then this sequence must also have a subsequent state s_j=[v',w',x',y'] where x'(y') = g,

and g is one of the expected messages at node v(w).

The second condition implies that if a machine reaches a receiving node where it must receive some message to progress further then this message will appear in its input channel in a finite time, and so the machine can progress. Therefore, a network is safe iff each machine can "progress infinitely often". A network can be proven safe using the technique of closed covers [9], as discussed later in Section 5.

Let (M,N) be a safe network. A node v in machine M is said to *occur infinitely often* in a communication sequence $<s_0,s_1,...>$ of (M,N) iff for any integer i there exists an integer j such that $j>i$ and s_j is of the form [v,w,x,y], for some w, x, and y. Similarly, we can define that a node in machine N occurs infinitely often in a communication sequence of (M,N).

Let (M,N) be a safe network. An edge e in machine M is said to *occur infinitely often* in a communication sequence $<s_0,s_1,...>$ of (M,N) iff for any integer i there exists an integer j such that $j>i$ and s_{j+1} follows s_j over e. Similarly, we can define that an edge in N occurs infinitely often in a communication sequence of (M,N).

Based on the above definitions, we can state that something "good" will occur infinitely often in a safe network (M,N) by stating that a node in M or in N (identified as being "good" or "useful" by the network designer) is guaranteed to occur infinitely often in every communication sequence of (M,N). In fact, the node needs not to occur infinitely often in every sequence. In particular, based on the assumption that both machines will progress fairly, the node needs not to occur in any sequence where one machine progresses in unfair fashion. Next we define three types of fair sequences.

A communication sequence q of a safe network (M,N) is called *weakly fair* iff the following condition is satisfied: For any node u in M or N, if u occurs infinitely often in q, then at least one outgoing edge of u occurs infinitely often in q.

A communication sequence q of a safe network (M,N) is called *fair* iff the following two conditions are satisfied for any node u, in M or N, that occurs infinitely often in q:

i. Each outgoing sending edge of u must occur infinitely often in q.

ii. If there is an infinite number of states of the form $[u,w_i,x_i,y_i]$ (or $[v_i,u,x_i,y_i]$) in q where message g is the head message in x_i (or y_i), and if u has an outgoing receiving edge e, labelled +g, then edge e must occur infinitely often in q.

A communication sequence q of a safe network (M,N) is called *strongly fair* iff the following condition is satisfied: For any node u in M or N, if u occurs infinitely often in q, then each outgoing edge of u occurs infinitely often in q.

Informally, a weakly fair sequence is one where each machine in the network is

forced to progress infinitely often. (Recall that the network is safe, and so each machine can indeed progress infinitely often.) A fair sequence is one where each machine is forced to execute infinitely often each edge that it can execute infinitely often. A strongly fair sequence is one where each machine is forced to execute infinitely often each edge that it reaches infinitely often.

The following lemma follows immediately from the above definitions:

Lemma 1 : Let (M,N) be a safe network,

i. Every fair sequence of (M,N) is weakly fair.

ii. Every strongly fair sequence of (M,N) is fair.

□

The following theorem states that every safe network must have weakly fair and fair sequences, but not necessarily strongly fair sequences.

Theorem 1 : Let (M,N) be a safe network.

i. (M,N) must have at least one weakly fair sequence.

ii. (M,N) must have at least one fair sequence.

iii. (M,N) may have no strongly fair sequence.

Proof : Since (M,N) is safe, proving parts i, ii, and iii can proceed as follows:

i. (M,N) must have a communication sequence $q=s_0,s_1,s_2,\ldots$ such that for $i=0,2,4,\ldots$, if s_{i+1} follows s_i over an edge in M (or N), then s_{i+2} follows s_{i+1} over an edge in N (or M). It is straightforward to show that this sequence is weakly fair.

ii. (M,N) must have a communication sequence $q=s_0,s_1,s_2,\ldots$ that can be constructed as follows:

   a. First, the initial state s_0 of (M,N) is added to q.

   b. After adding a state s=[v,w,x,y] to q, one of the outgoing edges, say e, of v or w should be selected for execution; then the state s' that follows s over e should be added to q. If such an s' exists, then e is said to be executed; otherwise, e cannot be executed at this time.

   c. To decide which of the outgoing edges should be selected for execution, the following priority scheme is adopted. Let k_e be the number of times an edge e in M or N is executed so far along q, on reaching a state

[v,w,x,y] along q, the outgoing edge e of v or w with the smallest k_e is selected for execution. If e cannot be executed at this time, then the outgoing edge e' of v or w with the next smallest $k_{e'}$ is selected, and so on. This scheme guarantees that if an edge can be executed infinite times along q, then it will be executed infinite times along q, i.e. q is fair.

iii. The proof is by a counter example: Network (M,N) in Figure 1b is safe and has exactly one communication sequence: $q=<[1,1,E,E], [2,1,E,g_1], [2,2,E,E], [2,1,g_2,E], [1,1,E,E], ...>$. This sequence is not strongly fair since node 1 in M occurs infinitely often in q while one of its outgoing edges, the one labelled $+g_3$, never occurs in q.

□

From Theorem 1, the existence of a strongly fair sequence is not guaranteed for a safe network. Nevertheless, when a strongly fair sequence does exist, one should be able to prove its existence. (Since this is part of proving weak liveness, as discussed in Section 4.) This motivates the need for a checkable sufficient condition that ensures the existence of a strongly fair sequence for a safe network. In order to state such a sufficient condition (in Theorem 2 below), we first introduce the concept of a communication subsequence.

A *communication subsequence* of a network (M,N) is a finite sequence $<r_0,...,r_k>$ of reachable states of (M,N) where there exists a communication sequence $<s_0,s_1,...>$ of (M,N), and there exists an integer j such that $r_i=s_{j+i}$, for i=0,...,k.

A node v in M is said to *occur* in a communication subsequence $<r_0,...,r_k>$ of a network (M,N) iff one of the states in the subsequence is of the form [v,w,x,y] for some w, x, and y. Similarly, we can define that a node in N occurs in a communication subsequence of (M,N).

An edge e in M is said to *occur* in a communication subsequence $<r_0,...,r_k>$ of a network (M,N) iff there exist two states r_i and r_{i+1} in the subsequence such that r_{i+1} follows r_i over e. Similarly, we can define that an edge in N occurs in a communication subsequence of (M,N).

Theorem 2 : If a network (M,N) has a communication subsequence $<r_0,...,r_k>$ such that the following conditions hold:

i. r_0 is the initial state of (M,N),

ii. there exists an i, $0\leq i<k$, such that $r_i=r_k$, and

iii. if a node occurs in the subsequence $<r_i,...,r_k>$, then every outgoing edge of

the node also occurs in the same subsequence,

then (M,N) has a strongly fair communication sequence.

Proof : Assume that (M,N) has a communication subsequence $<r_0,t,r_k,t',r_k>$ that satisfies conditions i, ii, and iii, where t and t' are finite subsequences of reachable states of (M,N). It is straightforward to show that the infinite sequence: $<r_0,t,r_k,t',r_k,t',r_k,...>$ is a strongly fair sequence of (M,N).

□

To show that the sufficient condition in Theorem 2 is "reasonable", the following lemma states that this condition is also necessary if the communication of the considered network is bounded.

Lemma 2 : Let (M,N) be a network whose communication is bounded. If (M,N) has a strongly fair sequence, then (M,N) must have a subsequence that satisfies conditions i, ii, and iii in Theorem 2.

Proof : Assume that (M,N) has a strongly fair sequence q. Since the communication of (M,N) is bounded, the number of reachable states is finite. Moreover, since q is an infinite sequence of reachable states, there is one reachable state r=[v,w,x,y] that occurs infinitely often in q. Select from q a finite prefix $<r_0,t,r>$ that satisfies the following three conditions:

i. r_0 is the initial state of (M,N).

ii. t is a finite subsequence of reachable states of (M,N).

iii. For any node u that occurs a finite number of times in q, all the occurrences of u in q must be in the finite prefix $<r_0,t,r>$. (In other words, any node that occurs in q after the finite prefix $<r_0,t,r>$ must occur infinitely often in q.)

Extend the finite prefix $<r_0,t,r>$ into a still finite, but possibly bigger, finite prefix $<r_0,t,r,t',r>$ such that the following two conditions are satisfied.

i. t' is a finite subsequence of reachable states of (M,N).

ii. For any node u that occurs in the subsequence $<r,t'>$, each outgoing edge of u must also occur in $<r,t'>$.

It is straightforward to show that the extended prefix $<r_0,t,r,t',r>$ satisfies conditions i, ii, and iii in Theorem 2.

□

Example 1 (Continues) :

Consider the safe network (M,N) in Figure 1a. The communication of (M,N) is bounded by 1. Hence, by generating and examining all reachable states, we can conclude that (M,N) is safe. By Theorem 1, this guarantees that (M,N) has weakly fair and fair sequences. Moreover, the following communication subsequence satisfies the three conditions in Theorem 2: <[1,4,E,E], [2,4,E,Ndata], [2,5,E,E], [2,4,Ack,E], [1,4,E,E], [2,4,E,Err], [2,6,E,E], [2,4,Nack,E], [3,4,E,E], [2,4,E,Err], [2,6,E,E], [2,4,Err,E], [3,4,E,E], [2,4,E,Ldata], [2,5,E,E], [2,4,Err,E], [3,4,E,E], [2,4,E,Ldata], [2,5,E,E], [2,4,Ack,E], [1,4,E,E]>. This implies that (M,N) also has a strongly fair sequence. As shown later the existence of this strongly fair sequence is useful in establishing the weak liveness of each node in network (M,N).

□

4. Liveness Properties

Based on the above three definitions of fair communication sequences, we present three degrees of node liveness. In what follows, let (M,N) be a safe network, and let u be a node in machine M or N.

Node u is said to be *weakly live* in (M,N) iff (i) (M,N) has at least one strongly fair sequence, and (ii) u occurs infinitely often in every strongly fair sequence of (M,N).

Node u is said to be *live* in (M,N) iff u occurs infinitely often in every fair sequence of (M,N).

Node u is said to be *strongly live* in (M,N) iff u occurs infinitely often in every weakly fair sequence of (M,N).

Notice that in the definitions of liveness and strong liveness, we did not insist on the existence of fair and weakly fair sequences (as we did in the definition of weak liveness), this is because the existence of these sequences is guaranteed by Theorem 1. The following two lemmas follow directly from the above definitions and Lemma 1.

Lemma 3 : Let (M,N) be a safe network and let u be a node in machine M or N.

i. If u is strongly live then u is live.

ii. If u is live and (M,N) has a strongly fair sequence, then u is weakly live.

□

Lemma 4 : Let (M,N) be a safe network, and let u_1 and u_2 be two nodes in the same machine, M or N, such that there is a directed path p from u_1 to u_2.

i. If u_1 is strongly live and all the edges in p are sending edges, then u_2 is live.

ii. If u_1 is live and network (M,N) has a strongly fair sequence, then u_2 is weakly live.

□

In [6], we show that the problem of "whether a node u is weakly live (live or strongly live) in a safe network (M,N)" is undecidable in general, we also characterize some special classes of networks for which the problem becomes decidable. In the current paper, we are interested in developing techniques to *prove* a positive answer for many instances of the problem. In other words, we are interested in sufficient conditions which can be checked easily and which, if satisfied by any instance of the problem, guarantee that indeed node u is weakly live (live or strongly live) in (M,N).

In the remainder of this section and in the next section, we discuss two classes of such sufficient conditions. The conditions discussed in this section are called *structure conditions* since they are conditions on the structures of the directed graphs of machines M and N. The conditions discussed in Section 5 are called *closed cover conditions* since they depend on the notion of closed covers discussed in [9]. Later we show that the closed cover conditions are more powerful than the structure conditions (Lemma 5 below); however, the structure conditions are in general easier to check than the closed cover ones.

The next two theorems, 3 and 4, state structure conditions that guarantee weak or strong liveness. (Using the structure condition for strong liveness in Theorem 4, along with Lemma 4 part i, one can establish liveness.)

Theorem 3 : Let (M,N) be a safe network that has a strongly fair sequence, and let u be a node in machine M (N). If there is a directed path from every node in M (N) to node u, then u is weakly live in (M,N).

Proof : It is sufficient to show that u occurs infinitely often in every strongly fair sequence of (M,N). Let q be any strongly fair sequence of (M,N). Since q is infinite, it must correspond to two infinite paths P and Q in machines M and N respectively. Since M (N) is finite, at least one node v in M(N) must occur infinitely often in P(Q). Since there is a directed path from v to u, then by induction on the number of edges in this path, we can show that u must occur infinitely often in q.

□

Theorem 4 : Let (M,N) be a safe network, and let u be a node in machine M(N). If every directed cycle in M(N) contains node u, then u is strongly live in (M,N).

Proof : It is sufficient to show that u occurs infinitely often in every weakly fair

sequence of (M,N). Let q be any weakly fair sequence of (M,N). Since q is infinite, it must correspond to two infinite paths P and Q in M and N respectively. Since both M and N are finite, each of P and Q must contain an infinite number of directed cycles in M and N respectively. Since each directed cycle in M(N) contains node u, then u must occur infinitely often in q.

□

Example 1 (Continues) : Consider the safe network (M,N) in Figure 1a. Using the structure conditions discussed in this section, we want to examine the liveness of each node in machine M.

i. Since each directed cycle in M contains node 2, then node 2 in M must be strongly live in (M,N), by Theorem 4.

ii. Since (M,N) has a strongly fair sequence as shown earlier, and since there is a directed path from every node in M to node 1(3) in M, then node 1(3) in M must be weakly live, by Theorem 3.

From ii, nodes 1 and 3 in M will be executed infinitely often provided that M and N do execute a strongly fair sequence. On the other hand, there is no guarantee that M and N will execute such a sequence. Instead of assuming (the rather severe assumption) that M and N will execute a strongly fair sequence, let us assume (the less restrictive assumption) that M and N will execute a fair sequence. As mentioned earlier, this new assumption has a natural interpretation, namely that the corrupting medium is "fair"; i.e. if an infinite number of messages are sent through this medium, then an infinite number of them will be corrupted and an infinite number of them will be transmitted without corruption. Proving that nodes 1 and 3 will be executed infinitely often under this new assumption is equivalent to proving that both nodes 1 and 3 are live. Unfortunately, the structure conditions discussed in this section cannot be used to prove that nodes 1 and 3 are live. This motivates the need for a new set of sufficient conditions.

□

5. Proving Liveness Using Closed Covers

The technique of closed covers is presented in [9] to prove that a network is safe. One advantage of this technique is that it can be used with networks whose communications are unbounded. (No other technique seems to be successful with such networks.) In this section, we extend this technique to prove node liveness. But first, a brief presentation of closed covers is in order.

A *closed cover* C for a network (M,N) is a set of states of (M,N) that satisfies the following four conditions:

i. The initial state of (M,N) is in C.

ii. Each directed cycle in the directed graph of M or N must have at least one node referenced in some state in C.

iii. The *acyclic version* AM of M with respect to C can be constructed from M by partitioning each node v, which is referenced in some state in C, into two nodes: One node, called the input version of v, has all the output edges of v and no input edges; the other node, called the output version of v, has all the input edges of v and no output edges. Similarly, the acyclic version AN of N with respect to C can be defined. The third condition can now be defined in terms of these acyclic versions. If the network (AM,AN) starts at a state s_1 in C and if it reaches a state s_2 after which no other state is reachable, then state s_2 must also be in C.

iv. The following condition should be satisfied for any state [v,w,x,y] in C, and for any two paths p and q ,in the acyclic versions AM and AN, that start with the input versions of v and w respectively, and terminate at the output versions of some nodes: Let s_i (r_i) be the sequence of sent (received) messages along path i, where i=p,q; then

$$\begin{array}{ll} \text{either} & [\ (x.s_q \prec r_p) \text{ and } (y.s_p \prec r_q)], \\ \text{or} & [\ \text{not}(x.s_q \prec r_p) \text{ and not}(y.s_p \prec r_q)], \end{array}$$

where

"." denotes the string concatenation operator, and

"$\prec$" denotes "is a proper prefix of".

A proof for the following theorem is in [9].

Theorem 5 : If a network has a closed cover, then it is safe.

□

Example 1 (Continues) : We show that the set C = {[1,4,E,E],[3,4,E,E]} is a closed cover for the network (M,N) in Figure 1a:

i. First, the initial state [1,4,E,E] of (M,N) is in C.

ii. Since nodes 1 and 3 in M and node 4 in N are referenced in C, every directed cycle in M or N has one node referenced in C.

iii. The acyclic versions AM and AN of M and N (respectively) with respect to C are shown in Figure 2a. If the network (AM,AN) starts at state [1,4,E,E], it must end its communication at [1,4,E,E] or at [3,4,E,E]; both are in C. Similarly, if (AM,AN) starts at state [3,4,E,E], it must end at either [1,4,E,E] or [3,4,E,E].

iv. Each state [v,w,x,y] in C is such that x=y=E. Also, any path in AM or AN,

that starts with the input version of some node and terminates at the output version of some node, has exactly one sending edge and one receiving edge. Therefore, for any two such paths p and q in AM and AN respectively, we have

$$\text{not}(s_q \prec r_p) \text{ and } \text{not}(s_p \prec r_q)$$

where

"$\prec$" denotes "is a proper prefix of", and

$s_i(r_i)$ is the sequence of messages sent (received) along path i, for i=p,q.

This completes the proof that C is a closed cover of (M,N), and so (M,N) is safe (by Theorem 5).

□

A closed cover C for a network (M,N) can be represented by a directed labelled graph G, called the *closed cover graph* of C, as follows:

i. Each state s in C is represented as a vertex, also labelled s, in G. (Notice that the "nodes" of G are called "vertices" to distinguish them from the "nodes" of machines M and N. For the same reason, the directed edges in G are called "arcs".)

ii. Let AM and AN be respectively the acyclic versions of M and N with respect to C. If the network (AM,AN) can reach from state s in C to state s' in C over a finite sequence $<e_0,e_1,...,e_r>$ of directed edges in M or N*. then there is a directed arc from the vertex labelled s to the vertex labelled s' in G; this arc is labelled with the set of edges $\{e_0,e_1,...,e_r\}$.

iii. No multiple arcs with identical labels are allowed in G.

Example 1 (Continues) : Figure 2b shows the closed cover graph G of the closed cover C=$\{[1,4,E,E], [3,4,E,E]\}$ for network (M,N) in Figure 1a. Notice that each directed edge e in M or N is defined in G by a tuple (i, j, k), where

i	is the source node of e,
j	is the label of e, and
k	is the destination node of e.

and each arc in G is labelled $\{e_0,e_1,...,e_r\}$.

□

Let C be a closed cover for a network (M,N), and let G be the closed cover graph of C. The vertex in G labelled with the initial state of (M,N) is called the *initial vertex* of G. A vertex (arc or directed cycle) in G is called *reachable* iff there is a directed path

---

*In other words, there exists a finite subsequence $<s_0,s_1,...,s_{r+1}>$ of states of (AM,AN) such that $s=s_0$, $s'=s_{r+1}$, and s_{i+1} follows s_i over e_i, i=0,1,...,r.

from the initial vertex of G to this vertex (arc or directed cycle). As an example, the vertex [1,4,E,E] in G of Figure 2b is its initial vertex; also each vertex, arc, and directed cycle in this G is reachable.

Let C be a closed cover for a network (M,N), and let G be the closed cover graph of C, also let u be a node in M or N, and e be a directed edge in M or N. Node u is said to *occur* in an arc of G iff the finite set that labels the arc contains an ingoing or outgoing edge of node u. Edge e is said to *occur* in an arc of G iff the finite set that labels the arc contains e. Node u or edge e is said to *occur* in a directed path (or cycle) in G iff it occurs in at least one arc in the path (or cycle). Node u or edge e is said to *occur infinitely often* in an infinite path in G iff it occurs in an infinite number of arcs in the path. Based on these concepts, we can now state, in the next theorem, a sufficient condition for a node to be weakly live.

Theorem 6 : Let (M,N) be a safe network that has a strongly fair sequence, and let C be a closed cover for (M,N), and G be the closed cover graph of C. Also let u be a node in M or N. If there is a directed path from every reachable vertex in G to an arc in which u occurs, then u is weakly live in (M,N).

Proof : Assume that there is a directed path from every reachable vertex in G to an arc in which u occurs, we show that u must occur infinitely often in every strongly fair sequence of (M,N). Let q be any strongly fair sequence of (M,N). Since q is strongly fair, it corresponds to two infinite directed paths P and Q in machines M and N respectively, such that every node occurrence in P or Q must also be in q, and vice versa. The two paths P and Q correspond to one infinite directed path p that starts with the initial vertex in G such that every node occurrence in P or Q must also be in p, and vice versa. It follows that any node in M or N occurs infinitely often in path p iff it occurs infinitely often in sequence q. Since p is an infinite path in a finite graph G, one of the vertices, say vertex n, in G must occur infinitely often in path p. Let vertex n be labelled with the state [v,w,x,y], therefore, nodes v and w occurs infinitely often in path p and so in sequence q. Since there is a directed path from vertex n to an arc in which u occurs, then there is a directed path either from v to u in M or from w to u in N (depending on whether u is in M or N, respectively). Since q is strongly fair and both v and w occur infinitely often in q, then u must occur infinitely often in q.

□

In order to state sufficient conditions for liveness and strong liveness (in Theorems 7 and 8 below), we need first to define basic and composite cycles in closed cover graphs and to introduce the concept of a message being sent in a composite cycle. This is done next.

Let C be a closed cover for a network (M,N), and let G be the closed cover graph of C. A reachable directed cycle L in G is called *basic* iff each vertex in G occurs at

most once in L. Cycle L is called *composite* iff it consists of one or more distinct basic cycles. For example, referring to the closed cover graph in Figure 2b, the self-loop at vertex [1,4,E,E] is basic. Also, the directed cycle that consists of:

1. the arc from vertex [3,4,E,E] to vertex [1,4,E,E],
2. the self-loop at vertex [1,4,E,E], and
3. an arc from vertex [1,4,E,E] to vertex [3,4,E,E]

is composite, since it consists of two basic cycles.

Let C be a closed cover for a network (M,N), and let G be the closed cover graph of C, and L be a composite cycle in G. A message g is said to be *sent by M(N) in L* iff one of the following two conditions holds,

i. There exists a sending edge labelled -g, in M(N), that occurs in L.

ii. One of the vertices in L is labelled with a state [v,w,x,y] where g is in y(x).

Based on these concepts, we can now state Theorems 7 and 8. The reader may find the statement of Theorem 7 complicated at first, but the first paragraph in its proof provides some useful insight about the theorem.

Theorem 7 : Let (M,N) be a safe network, and let C be a closed cover for (M,N) and G be the closed cover graph of C. Also let u be a node in M or N. u is live in (M,N), if for each reachable directed composite cycle L in G, one of the following three conditions holds:

i. u occurs in L.

ii. There is a node in M or N, that occurs in L, but one of its outgoing sending edges does not occur in L.

iii. For every message g sent by M(N) in L, there exists a node d, in N(M), that occurs in L such that d has an outgoing edge labelled +g. Moreover, for some message g sent by M(N) in L, there is no edge labelled +g, in N(M), that occurs in L.

Proof : Before we prove the theorem formally, let us sketch the proof informally. For node u to be live, it should occur infinitely often in every fair sequence of (M,N). Any fair sequence q of (M,N) must correspond to an infinite path p, that starts from the initial vertex in G. Since p is infinite and G is finite, a maximal composite cycle L_{max} in G must be repeated in p infinite times. As shown later, such an L_{max} cannot satisfy conditions ii and iii in the theorem; hence it must satisfy condition i. Therefore, node u must occur in L_{max}, and so it must occur in path p and in sequence q infinitely often; this establishes the liveness of node u. (The exact reasoning for L_{max} not to satisfy ii and

iii is discussed later in detail; we give here some hints: If L_{max} satisfies condition ii, then some node in M or N will occur in sequence q infinitely often but one of its outgoing sending edges will not occur in q infinitely often; this contradicts the assumption that q is fair. If L_{max} satisfies condition iii, then (a) sequence q must have an infinite number of states of the form $[v,w_i,x_i,y_i]$, where v has an outgoing edge e labelled +g and the head message in x_i is g, and (b) edge e does not occur infinitely often in q. These conditions contradict the fairness of q.)

Assume that each reachable directed composite cycle in G satisfies either one of i, ii and iii above, we show that u must occur infinitely often in every fair sequence of (M,N). Let q be any fair sequence of (M,N). Since q is fair, it corresponds to two infinite directed paths P and Q in machines M and N respectively such that every node occurrence in P or Q must also be in q, and vice versa. The two paths P and Q correspond to one infinite directed path p in G, such that p starts with the initial vertex in G, and every node occurrence in P or Q must also be in p, and vice versa. It follows that any node in M or N occurs infinitely often in path p iff it occurs infinitely often in sequence q.

Since p is an infinite path in a finite graph G, a finite number of basic cycles in G must occur infinitely often in p, let these basic cycles be L_1, L_2, ..., L_m. Let L_{max} be any composite cycle that consists of L_1, L_2, ..., and L_m; it is straightforward to show that a node (or an edge) in M or N occurs in L_{max} iff it occurs infinitely often in path p (and in sequence q).

There are two cases to consider:

a. u occurs in L_{max}, in which case u occurs infinitely often in sequence q.

b. u does not occur in L_{max}, then there are two possibilities, and we show that each of them leads to a contradiction,

    1. L_{max} satisfies condition ii. There is a node, say v, in M or N which occurs in L_{max}, but one of its outgoing sending edges does not occur in L_{max}. Therefore, v occurs infinitely often in q, but one of its outgoing sending edges does not occur infinitely often in q. This contradicts the assumption that q is a fair sequence.

    2. L_{max} satisfies condition iii. From condition iii, for every message g sent by M(N) in L_{max}, there exists a node d, in N(M), that occurs in L_{max} such that d has an outgoing edge labelled +g. (Notice that this outgoing edge is not necessarily in L_{max}.) This implies that every message g sent by M(N) along q must be received by N(M) along q. (Otherwise, let g' be the first message sent by M(N) along q and never received along q. Message g' must be sent in L_{max}, then there exists a node d', in N(M), that occurs in L_{max} such that d' has an

outgoing edge labelled +g'. Node d' must occur in q infinitely often, i.e. there is an infinite number of states of the form $[d',w_i,g'.x_i,y_i]$ (or $[v_i,d',x_i,g'.y_i]$), where v_i and w_i are any nodes in M or in N respectively, and x_i and y_i are any strings of messages. Since q is fair, this message g' must be received along q.) Also from condition iii, for some message g sent by M(N) in L_{max}, there is no edge labelled +g, in N(M), that occurs in L_{max}. Since g is sent by M(N) in L_{max}, then one of the following two conditions holds:

i. There is a sending edge labelled -g, in M(N), that occurs in L_{max}. This edge must occur infinitely often in q, i.e. message g must be sent by M(N) infinitely often along q, and so must be received by N(M) infinitely often along q. This implies that an edge labelled +g, in N(M), must occur in L_{max}, contradiction.

ii. There is no sending edge labelled -g, in M(N), that occurs in L_{max}; instead, L_{max} has a vertex labelled [v,w,x,y], where the y-component (x-component) contains g. Since no edge labelled +g, in N(M), occurs in L_{max}, then the y-component (x-component) of each vertex in L_{max} must contain g. If a vertex in L_{max} is reached along q, then message g must have been sent previously by M(N). This message must be received by a receiving edge that does not occur in L_{max}, but later, q must return to a vertex in L_{max} (Since each vertex in L_{max} must be reached infinitely often along q). But by reaching this vertex, a message g must have been sent earlier and so must be received later by a receiving edge that does not occur in L_{max}, and so on. This means that some (receiving) edge that does not occur in L_{max} must occur infinitely often along q, contradiction.

□

Theorem 8 : Let (M,N) be a safe network, and let C be a closed cover for (M,N), G be the closed cover graph of C. Also let u be a node in M or N. If each reachable directed basic cycle in G has at least one arc in which u occurs, then u is strongly live in (M,N).

Proof : Assume that each reachable directed basic cycle in G has at least one arc in which u occurs, we show that u must occur infinitely often in every weakly fair sequence of (M,N). Let q be any weakly fair sequence of (M,N). Since q is weakly fair, then it corresponds to two infinite directed paths P and Q in machines M and N respectively such that every node occurrence in P or Q must also be in q, and vice versa. The two paths P and Q correspond to one infinite directed path p that starts with the initial vertex in G such that every node occurrence in P or Q must also be in p, and vice versa.

It follows that any node in M or N occurs infinitely often in path p iff it occurs infinitely often in sequence q. Since p is an infinite path in a finite graph G, p must include an infinite number of reachable directed basic cycles of G. Since node u occurs in each of these cycles, then it must occur infinitely often in path p. Therefore, u must occur infinitely often in the weakly fair sequence q.

□

Example 1 (Continues) : Let us apply Theorems 6, 7, and 8 to establish liveness of the different nodes of machine M in network (M,N) in Figure 1a. Referring to the closed cover graph G of this network in Figure 2b, we observe the following:

i. Each vertex in G has at least one outgoing arc in which node 1(3) in M occurs. Hence node 1(3) in M is weakly live in (M,N) by Theorem 6.

ii. To show that node 1 in M is live in (M,N) by Theorem 7, the following steps should be followed:

   a. Remove from G all the arcs where node 1 occurs.

   b. This leaves only the three self-loops at vertex [3,4,E,E]; each self-loop constitutes one basic cycle.

   c. From these three basic cycles, construct seven composite cycles, and show that each of them satisfies condition ii in Theorem 7.

iii. To show that node 3 in M is live in network (M,N) by Theorem 7, we follow the same three steps in ii, except that this time there remains only one basic cycle, namely the self-loop at node [1,4,E,E], and so only one composite cycle needs to be considered.

iv. Node 2 in M occurs in every arc in G, hence it is strongly live by Theorem 8.

□

Theorems 6 and 8 state some closed cover conditions that can establish weak and strong node liveness in some networks. Weak and strong node liveness can also be established using the structure conditions in Theorems 3 and 4. It is interesting to compare the cost and effectiveness of these two sets of conditions:

i. Examining the structure conditions requires only to examine the directed graphs of the two machines in the network. This takes a time O(e), where e is the number of edges in the two machines. By contrast, examining the closed cover conditions requires to construct and examine a closed cover graph for the network. In the worst case, this may require an exponential time with respect to e.

ii. As shown in Lemma 5 below, the closed cover conditions in Theorems 6 and 8 are *more powerful* than the structure conditions in Theorems 3 and 4. This

means that there is a node in some network which can be proven weakly (strongly) live using the conditions in Theorem 6 (8), but which cannot be proven so using the conditions in Theorem 3 (4). (Whether the conditions in Theorems 3 and 4 are also more powerful than those in Theorems 6 and 8 remains an unanswered question.)

From these two observations, one should always try first to use the structure conditions in Theorems 3 and 4 to establish node liveness. Only when these conditions fail, should one resort to the closed cover conditions in Theorems 6 and 8.

Lemma 5 : The closed cover conditions in Theorems 6 and 8 are more powerful than the structure conditions in Theorems 3 and 4 respectively.

Proof : Consider the network (M,N) in Figure 3a. Node 2 in M does not satisfy the sufficient condition in Theorem 3, therefore, its weak liveness cannot be established using this theorem. On the other hand, it is straightforward to check that the set {[1,4,E,E], [2,5,E,E], [3,6,E,E]} is a closed cover for this network, and so its closed cover graph is as shown in Figure 3b. Using this closed cover graph, node 2 in M can be proven weakly live by Theorem 6. Similarly, the strong liveness of node 2 in M can be established by Theorem 8, but not by Theorem 4.

□

The closed cover conditions in Theorems 6, 7, and 8 are applicable even if the considered closed cover C for network (M,N) is infinite, i.e. C contains an infinite number of states of (M,N). However, in this case the closed cover must have a *finite representation* so that its closed cover graph remains finite as required by the proofs of Theorems 6, 7, and 8. One finite representation of an infinite closed cover is as follows:

$C=\{[v_1,w_1,X_1,Y_1], ..., [v_r,w_r,X_r,Y_r]\}$ where

v_i (i=1,...,r)	is a node in machine M,
w_i (i=1,...,r)	is a node in machine N,
X_i (i=1,...,r)	is a (possibly infinite) set of message strings, and
Y_i (i=1,...,r)	is a (possibly infinite) set of message strings.

Each four-tuple $[v_i,w_i,X_i,Y_i]$, called a state schema of (M,N), represents a (possibly infinite) set of states of (M,N), each state is of the form $[v_i,w_i,x_i,y_i]$ where x_i is a string in set X_i and y_i is a string in set Y_i. In this case, each vertex in the corresponding closed cover graph G represents one state schema in C. Therefore, G is finite (i.e. has r vertices) and Theorems 6, 7, and 8 are still applicable. However, condition iii in Theorem 7 should be restated as follows:

iii. For every message g where there is a vertex labelled [v,w,X,Y] in L such that g is in a string in Y(X), there exists a node d, in N(M), that occurs in L such that d has an

outgoing edge labelled +g. Moreover, there is a message g such that there is no edge labelled +g, in N(M), that occurs in L, and one of the following two conditions holds:

a. There is a sending edge labelled -g in M(N), that occurs in L.

b. One of the vertices in L is labelled with [v,w,X,Y] where g is in every string in Y(X).

6. Examples

In this section, we discuss three practical protocol examples and prove their node liveness using the previous results. There are three objectives of this exercise. The first objective is to illustrate (in Examples 2 and 3) the applicability of our results to unbounded networks, i.e. networks where the number of reachable states is infinite. Second, we want to discuss (in Example 3) how to construct and deal with infinite closed covers. Third, we demonstrate (in Example 4) that our techniques are applicable to networks with any number of machines not just two machines.

Example 2 (A Start-stop Protocol) : The Start-stop protocol was developed to transmit data characters from a sender to a receiver over an asynchronous serial line [17]. The line remains idle so long as the sender does not wish to send. For the sender to send one data character, it follows the next steps:

i. First, the sender sends a start bit to indicate to the receiver its intention of sending one data character.

ii. Then, the sender sends the character bit by bit to the receiver.

iii. Finally, the sender sends two stop bits to indicate the end of transmission. The line returns to its idle state until the next character is sent.

This protocol is modeled by the two machines M and N in Figure 4a, where M models the sender and N models the receiver, and the sent messages have the following meanings:

Idle	indicates to the receiver that the line is in an idle state.
Start	denotes the starting bit.
One, Two, Three, Four, and Five	denote the five bits in a data character.
Stop	denotes a stop bit.

Although this network seems simple, its communication is unbounded (i.e. the number of its reachable states is infinite); hence, proving its safety and liveness is not straightforward. Nevertheless, we use Theorem 7 to show that each node in this network is indeed live. (Notice that the fairness assumption in this case is equivalent to the reasonable assumption that data characters will be transmitted over the line infinitely often, and that idle periods of the line will occur infinitely often.)

It is straightforward to show that {[1,1,E,E]} is a closed cover for the network. (Notice that although the network is unbounded, its closed cover is finite; in the next example, we discuss an unbounded network whose closed cover is infinite.) The corresponding closed cover graph is shown in Figure 4b. It has two self-loops, labelled A and B. The self-loop labelled A satisfies condition ii in Theorem 7, and each of the self-loop labelled B and the composite cycle that consists of the two self-loops satisfies condition i in Theorem 7, for each node in M or N. Therefore, each node of M or N is live.

Notice that the liveness of some of these nodes (not all of them) can be also established using the structure condition in Theorem 4 along with Lemma 3 part i and Lemma 4 part i.

Example 3 (An Alternating Bit Protocol) : The Alternating Bit protocol was proposed [1] to ensure reliable transmission of data messages from a sender to a receiver over a communication medium that can corrupt or lose transmitted messages. Since we have already discussed (in Example 1) a medium that can corrupt messages, we discuss in the current example a version of the Alternating Bit protocol where the medium can only lose messages. This version of the protocol can be defined as follows.

i. The sender sends its data messages, one by one, to the receiver; but after sending each data message, it must wait to receive an acknowledgement before sending the next data message.

ii. The sender is provided with a timeout capability to resend its last data message, if it does not receive any acknowledgement for some time.

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

Machines M and N in Figure 5a model the sender and the receiver respectively. Instead of modeling the medium as a separate machine, the medium's effect is modeled as follows. Whenever a machine, M or N, sends a message g, it either sends g or sends a special message L that denotes a lost message; on receiving L, the other machine must remain in its current state. The other exchanged messages between M and N have the following meanings:

D_i (i=0,1) denotes a data message with a bit of value i attached to it.
A_i (i=0,1) denotes the acknowledgement of D_i.

Notice that the timeout capabilities in sender M are modeled by (i) the self-loop labelled $-D_0$ at node 2, and (ii) the self-loop labelled $-D_1$ at node 4.

Proving liveness of this protocol consists of proving that node 1 (where new D_0 data messages are sent), and node 3 (where new D_1 data messages are sent) are both live. (Recall that if these nodes are live then they will be executed infinitely often provided that the machines behave fairly. The fair behaviour assumption amounts in this case to assuming that if an infinite number of messages are sent through the medium, then infinite number of them will be delivered without loss.) Next, we sketch a proof that node 1 is live; a similar proof can establish the liveness of node 3.

It is straightforward to show that the following set is a finite representation of a closed cover C for network (M,N):

$$
\begin{aligned}
C = \{ & [1,5,(A_1+L)^k,(D_1+L)^k],\ [1,8,(A_1+L)^k,(D_1+L)^k], \\
& [2,5,(A_1+L)^k,(D_1+L)^m(D_0+L)^{k-m}],\ [2,6,(A_1+L)^m(A_0+L)^{k-m},(D_0+L)^k], \\
& [2,7,(A_1+L)^m(A_0+L)^{k-m},(D_0+L)^k],\ [2,8,(A_1+L)^k,(D_1+L)^m(D_0+L)^{k-m}], \\
& [3,6,(A_0+L)^k,(D_0+L)^k],\ [3,7,(A_0+L)^k,(D_0+L)^k], \\
& [4,5,(A_0+L)^m(A_1+L)^{k-m},(D_1+L)^k],\ [4,6,(A_0+L)^k,(D_0+L)^m(D_1+L)^{k-m}], \\
& [4,7,(A_0+L)^k,(D_0+L)^m(D_1+L)^{k-m}],\ [4,8,(A_0+L)^m(A_1+L)^{k-m},(D_1+L)^k]\}
\end{aligned}
$$

(The two variables k and m can take any of the values 0,1,2,... such that $k \geq m$.)

A portion of the closed cover graph G of C is shown in Figure 5b. The labels of the shown arcs are as follows:

$U_1=\{(2,-L,2),(5,+L,5)\}$ $U_2=\{2,-D_0,2),(5,+L,5)\}$
$U_3=\{(2,+L,2),(5,+L,5)\}$ $U_4=\{(2,+A_1,2),(5,+L,5)\}$
$V_1=\{(2,-L,2),(8,-L,5)\}$ $V_2=\{(2,-L,2),(8,-A_1,5)\}$
$V_3=\{(2,-D_0,2),(8,-L,5)\}$ $V_4=\{(2,-D_0,2),(8,-A_1,5)\}$
$V_5=\{(2,+L,2),(8,-L,5)\}$ $V_6=\{(2,+L,2),(8,-A_1,5)\}$
$V_7=\{(2,+A_1,2),(8,-L,5)\}$ $V_8=\{(2,+A_1,2),(8,-A_1,5)\}$
$W_1=\{(2,-L,2),(5,+D_1,8)\}$ $W_2=\{(2,-D_0,2),(5,+D_1,8)\}$
$W_3=\{(2,+A_1,2),(5,+D_1,8)\}$ $W_4=\{(2,+L,2),(5,+D_1,8)\}$
$X_1=\{(2,-L,2),(6,-A_0,7)\}$ $X_2=\{(2,-L,2),(6,-L,7)\}$
$X_3=\{(2,-D_0,2),(6,-A_0,7)\}$ $X_4=\{(2,-D_0,2),(6,-L,7)\}$
$X_5=\{(2,+A_1,2),(6,-A_0,7)\}$ $X_6=\{(2,+A_1,2),(6,-L,7)\}$
$X_7=\{(2,+L,2),(6,-A_0,7)\}$ $X_8=\{(2,+L,2),(6,-L,7)\}$
$Y_1=\{(2,-L,2),(7,+D_0,6)\}$ $Y_2=\{(2,-D_0,2),(7,+D_0,6)\}$
$Y_3=\{(2,+A_1,2),(7,+D_0,6)\}$ $Y_4=\{(2,+L,2),(7,+D_0,6)\}$
$Z_1=\{(2,-L,2),(7,+L,7)\}$ $Z_2=\{(2,-D_0,2),(7,+L,7)\}$
$Z_3=\{(2,+L,2),(7,+L,7)\}$ $Z_4=\{(2,+A_1,2),(7,+L,7)\}$

To show that node 1 is live, delete from the closed cover graph all the arcs where

node 1 occurs, (this leaves only those solid arcs shown in Figure 5b), and show that any remaining composite cycle must either satisfy condition ii or iii in Theorem 7. It is straightforward to show that any of the composite cycles in Figure 5b satisfies either of the two conditions.

Example 4 (A Carrier Sense Multiple Access Protocol with Collision Detection - CSMA/CD) : The CSMA/CD protocol is used for local area networks where many stations are connected to a single communication medium [19]. At any instant, each station can start a transmission over the medium to some other station(s). However, if two or more stations start transmitting at about the same time (they are said to collide), then each of them detects the collision and backs off for some time before trying retransmission again.

We model this protocol by a network (A,B,M) of three communicating finite state machines A, B, and M shown in Figure 6. Machines A and B model two identical stations connected to the medium, while M models the medium itself. The network operates indefinitely in successive stages. The following activities occur in a typical stage:

i. Each station A or B sends to medium M either a "Rqst" message (indicating that it intends to send a sequence of data frames in the current stage), or a "Norqst" message (indicating that it does not intend to send data frames at the current stage).

ii. If both stations send "Norqst" messages, the medium responds by sending an "Off" (Carrier-Off signal) message to each of them. Both stations and the medium return to their initial nodes (not necessarily at the same time), and the next stage starts.

iii. If both stations send "Rqst" messages, the medium responds by sending a "Collision" message to each of them indicating that neither station can send data frames at the current stage. Both stations and the medium return to their initial nodes, and the next stage starts.

iv. If one station, say A, sends a "Norqst" message while the other station, B, sends a "Rqst" message, then the medium sends to A an "On" (Carrier-On signal) message, followed by a sequence of data frames from B. Finally the medium receives an "Off" message from B, then sends an "Off" message to A and all machines return to their initial nodes, and the next stage starts.

It is straightforward to show that the following set C={[1,1,6,E,E,E,E], [3,5,12,E,E,E,E], [5,3,18,E,E,E,E]} is a closed cover for the network (A,B,M). This guarantees that the network is safe.

Proving liveness of this network consists of showing that both nodes 5, where data frames are sent, in stations A and B will be executed infinitely often. One possibility is to show that both nodes are live (i.e. if each machine behaves fairly, then the two nodes

will be executed infinitely often). Unfortunately, this is not the case; for instance, neither node will be executed at all along the *fair* sequence where both machines send "Norqst" in stages 1,3,5,7,..., and both machines send "Rqst" in stages 2,4,6,8,... .

The only remaining possibility now is to show that both nodes 5 are weakly live. Indeed, using the structure condition in Theorem 3 (which can be extended to networks with any number of communicating machines), one can show that both nodes 5 in machines A and B are weakly live.

Weak liveness of the two nodes guarantees that those two nodes will be executed infinitely often *only if* the three machines execute a strongly fair sequence. One way to force the three machines to execute a strongly fair sequence is by using randomization [19]. Whenever a station, A or B, returns to its initial node (node 1), it should flip a fair coin to decide whether to send a "Rqst" or "Norqst" message in the next stage. It is straightforward to show that if the two coins are fair and independent, then the three machines will execute a strongly fair sequence along which both nodes 5 in stations A and B will be executed infinitely often.

7. Concluding Remarks

The main result in this paper is a set of sufficient conditions (Theorems 6, 7, and 8) that can be used to establish three degrees of node liveness in safe networks of communicating finite state machines. An interpretation of these different degrees of node liveness is discussed next.

Any safe network can be completely defined by a (possibly infinite) set of communication sequences, where each sequence defines one possible infinite history of the communication between the different machines in the network. Although the network is capable of executing any one of these sequences, it ultimately ends-up executing only one of them. Therefore, there are two alternatives to establish that a node in one of the machines is live (i.e. will be executed infinitely often during the course of communication):

Either i. show that the node will be executed infinitely often along every communication sequence of the network,

or ii. show that the node will be executed infinitely often along the one sequence which the network ends-up executing.

Clearly, the first alternative is very restrictive; only the second one seems applicable to a wide variety of interesting networks (that model "real" protocols). The second alternative can be divided into two parts:

A. First, show that the node under consideration will be executed infinitely often along every sequence in some class (of sequences).

B. Then, show how to "restrict" the behaviour of each machine in the network so that the resulting communication sequence of the network is guaranteed to fall into this same calss.

In this paper, we have identified three classes of communication sequences, namely weakly fair, fair, and strongly fair sequences. The relationships between these three classes can be represented by the Venn diagram in Figure 7 (See Lemma 1). Next we discuss how to solve parts A and B for each of these three classes.

Weakly Fair Sequences: The sufficient conditions in Theorems 3 and 6 can be used to solve A, and establish the strong liveness of the node under consideration. To solve B, each machine in the network should be restricted (or forced) to progress infinitely often. (Recall that the network is safe, and so each machine can indeed progress infinitely often.)

Fair Sequences: The sufficient conditions in Theorem 7 can be used to solve A, and establish the liveness of the node under consideration. To solve B, each machine in the network should be restricted to be fair with respect to selecting its next action. (In the proof of Theorem 1 part ii, we described a procedure by which each machine can ensure a fair selection of its next actions.) Notice that, as shown in Examples 1 and 3, if a medium that can corrupt or lose messages is modeled as one machine, then this restriction merely corresponds to the familiar assumption of "fair medium".

Strongly Fair Sequences: The sufficient conditions in Theorems 4 and 8 can be used to solve A, and establish the weak liveness of the node under consideration. One way to solve B is by constructing a finite representation of a strongly fair sequence (as discussed in the proof of Theorem 2). This finite representation is then provided to each machine in the network, and each of them is restricted to behave according to it. In some cases, randomization can be also used to solve B as discussed in Example 4.

Notice that the sufficient conditions in Theorems 6, 7, and 8 can be checked algorithmically for any given network provided that the next scenario is followed:

i. First, the network's designer provides a set of states as a closed cover for the network under consideration.

ii. Whether this set is indeed a closed cover for the given network can be checked algorithmically. If the check fails, the designer is required to provide a correct closed cover, and so on. This continues until a closed cover for the network is established.

iii. Finally, the closed cover graph is generated, and the sufficient conditions in Theorems 6, 7, and 8 can be checked algorithmically.

Although most of the discussion in this paper has centered around networks with two communicating machines, the results can be extended in a straightforward manner

to networks with r machines ($r \geq 2$). (See for instance Example 4.) Also, the notion of node liveness in this paper can be extended in a straightforward fashion to liveness of edges and to liveness of sets of nodes and/or edges.

The discussion in this paper suggests the following interesting question. Are the three classes of communication sequences discussed in this paper "sufficient" for most practical networks, or are there other interesting classes of sequences? This question is still open, and seems attractive for further research. Another interesting question is whether the sufficient conditions in Theorems 3, 4, 6, 7, and 8 are "reasonable" or more relaxed sufficient conditions are needed in some cases.

REFERENCES

1. Bartlett, K. A., R. A. Scantlebury, and P. T. Wilkinson, "A note on reliable full-duplex transmission over half-duplex links," *Comm. ACM*, Vol. 12, May 1969, pp.260-261.

2. Brand, D. and P. Zafiropulo, "On communicating finite-state machines," *Journal ACM*, Vol. 30, No. 2, April 1983, pp. 323-342.

3. Bochmann, G. V. "Finite state description of communication protocols," *Computer Networks*, Vol. 2, 1978, pp. 361-371.

4. Bochmann, G. V. and C. Sunshine, "Formal methods in communication protocol design," *IEEE Trans. on Commun.*, April 1980, pp.624-631.

5. Bochmann, G. V. et al, "Experience with formal specifications using an extended state transition model," *IEEE Trans. on Commun.*, Dec. 1982, pp. 2506-2513.

6. Chang, C. K. and M. G. Gouda, "Deciding liveness for special classes of communicating finite state machines," In preparation.

7. Chow, C. H., M. G. Gouda, and S. S. Lam, "An exercise in constructing multi-phase communication protocols," In *Proc. of SIGCOMM'84 Symposium*, June 1984.

8. Gouda, M. G., "An example for constructing communicating machines by step-wise refinement," in *Proc. 3rd IFIP Workshop on Protocol Specification, Testing, and Verification*, edited by H. Rudin and C. H. West, North-Holand, 1983, pp. 63-74.

9. Gouda, M. G., "Closed covers: to verify progress for communicating finite state machines," *TR-191*, Dept. of Computer Sciences, Univ. of Texas at Austin, Jan. 1982. Revised Jan. 1983. To appear in *IEEE Trans. on Software Engineering*.

10. Gouda, M. G., E. G. Manning, and Y. T. Yu, "On the progress of communication between two finite state machines," *TR-200*, Dept. of

Computer Sciences, Univ. of Texas at Austin, May 1982. Revised Oct. 1983.

11. Gouda, M. G. and L. E. Rosier, "Communicating finite state machines with priority channels," In *Proc. of the Eleventh International Colloqium on Automata, Languages, and Programming (ICALP)*", 1984.

12. Gouda, M. G. and Y. T. Yu, "Maximal progress state exploration," In *Proc. of SIGCOMM'83 Symposium*, 1983, pp. 68-75.

13. Gouda, M. G. and Y. T. Yu, "Protocol validation by maximal progress state exploration", *TR-211*, Dept. of Computer Sciences, Univ. of Texas at Austin, July 1982. To appear in *IEEE Trans. on Commun.* Jan. 1984.

14. Gouda, M. G. and Y. T. Yu, "Synthesis of communicating finite state machines with guaranteed progress", *TR-179*, Dept. of Computer Sciences, Univ. of Texas at Austin, June 1981. Revised Jan. 1983. Revised Oct. 1983. To appear in *IEEE Trans. on Commun.*, 1984.

15. Hailpern, B. T. and S. S. Owicki, "Modular verification of computer communication protocols," *IEEE Trans. on Commun.*, Vol. com-31, No. 1, Jan. 1983, pp. 56-68.

16. Lam, S. S. and A. U. Shankar, "Protocol verification via projections," *TR-207*, Dept. of Computer Sciences, Univ. of Texas at Austin, August, 1982. To appear in *IEEE Trans. on Software Engineering.*

17. McNamara, J. E., "Technical aspects of data communication," Digital Equipment Corp., Maynard, Mass., 1977.

18. Merlin, P. M. and G. V. Bochmann, "On the construction of submodule specifications and communication protocols," *ACM Trans. on Programming Languages and Syst.*, Vol. 5, No. 1, Jan. 1983, pp. 1- 25.

19. Metcalfe, R. M. and D. R. Boggs, "Ethernet: distributed packet switching for local computer networks," *Comm. ACM*, Vol. 19, No. 7, July 1976, pp. 395-404.

20. Misra, J. and K. M. Chandy, "Proof of networks of processes," *IEEE Tran. on Software Eng.*, Vol. SE-7, No. 4, July 1981.

21. Misra, J., K. M. Chandy and T. Smith, "Proving safety and liveness of communicating processes with examples," in *Proc ACM SIGACT-SIGOPS Symposium on Principls of Distributed Computing*, Aug. 1982, pp. 18-20.

22. Owicki, S. and L. Lamport, "Proving liveness properties of concurrent programs," *ACM Trans. on Programming Languages and Syst.*, Vol. 4, No. 3, July 1982, pp. 455-495.

23. Pnueli, A., "The temporal logic of programs," in *Proc. 18th Annual Symposium on Foundations of Computer Science*, Providence, RI, Oct.

1977, pp. 46-57.

24. Pnueli, A., "The temporal semantics of concurrent programs," *Theoretical Computer Science*, Vol. 13, 1981, pp. 45-60.

25. Razouk, R. R. and G. Estrin, "Modeling and verification of communication protocols in SARA: The X.21 interface," *IEEE Trans. on Comput.*, Vol. C-29, No. 12, Dec. 1980, pp. 1038-1052.

26. Rosier, L. E. and M. G. Gouda, "Deciding progress for a class of communicating finite state machines," *TR-83-22*, Dept. of Computer Sciences, Univ. of Texas at Austin, Oct. 1983. Submitted for publication.

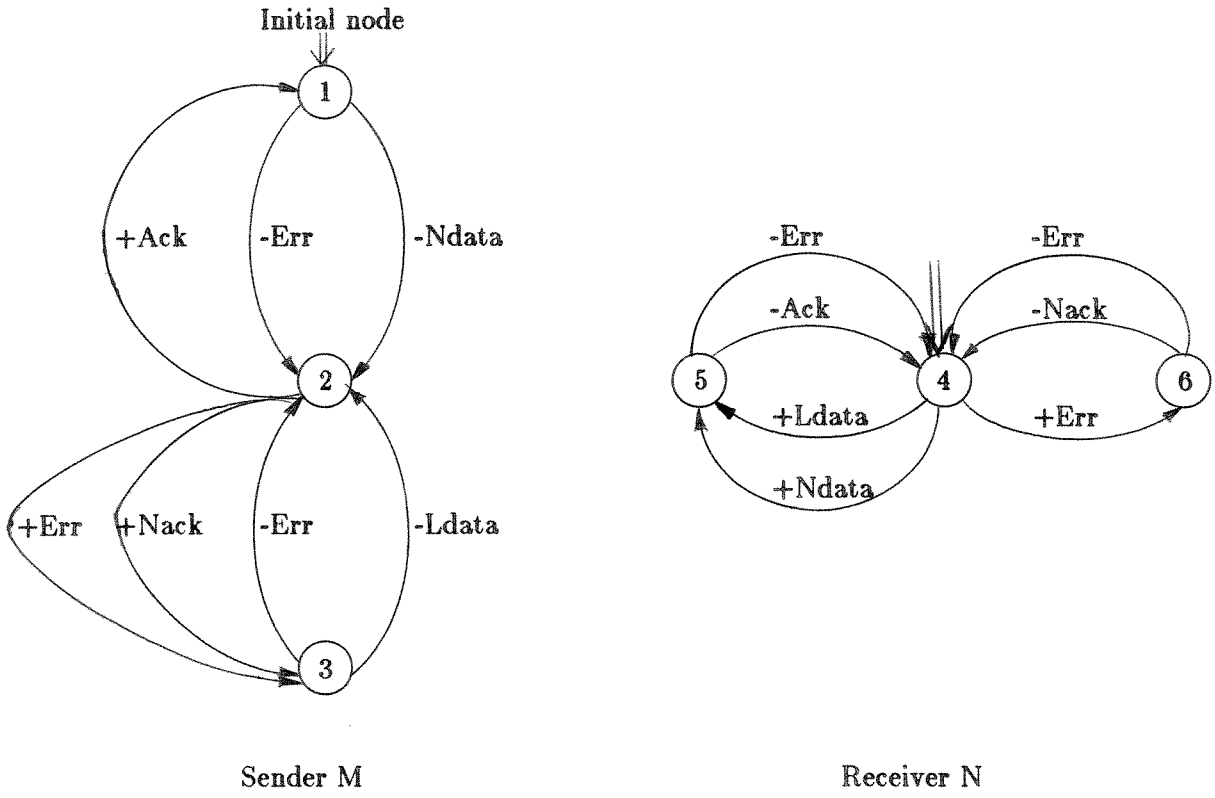
27. Rubin, J. and C. H. West, "An improved protocol validation technique," *Computer Networks*, Vol. 6, No. 2, April 1982.

28. West, C. H. and P. Zafiropulo, "Automated validation of a communications protocol: The CCITT X.21 recommendations," *IBM J. Res. Develop.*, Vol. 22, Jan. 1978, pp. 60-71.

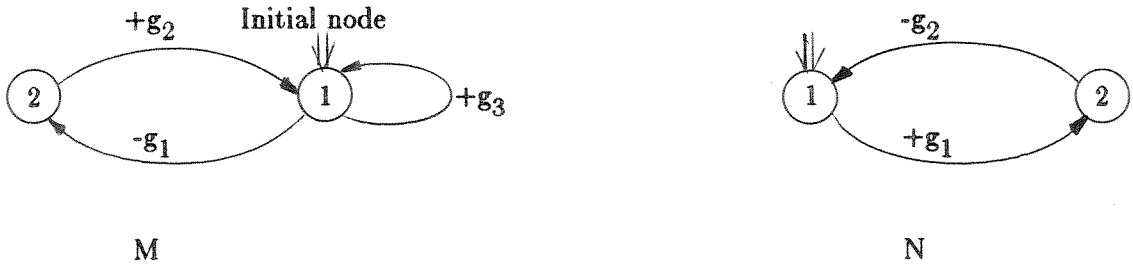
29. Yu, Y. T. and M. G. Gouda, "Deadlock detection for a class of communicating finite state machines," *IEEE Trans. on Commun.*, Dec. 1982, pp. 2514-2519.

30. Yu, Y. T. and M. G. Gouda, "Unboundedness detection for a class of communicating finite-state machines," *Information Processing Letters*, Vol. 17, Dec. 1983, pp. 235-240.

31. Zafiropulo, P., C. H. West, H. Rudin, D. Brand, and D. Cowan, "Towards analyzing and synthesizing protocols," *IEEE Trans. on Commun.*, Vol. COM-28, No. 4, April 1980, pp. 651-661.

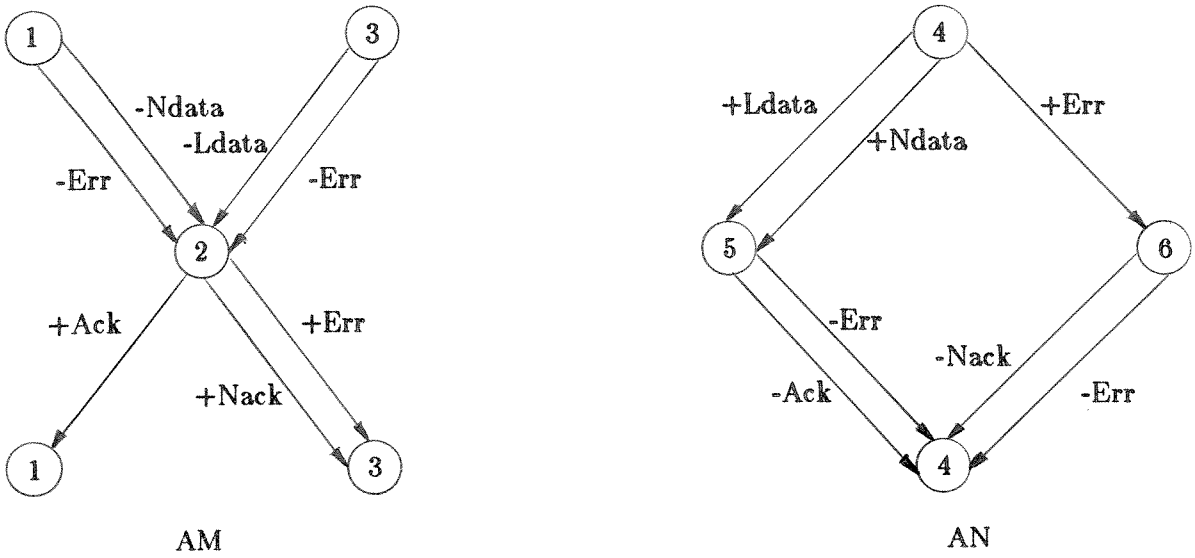


(a) A network for Example 1.

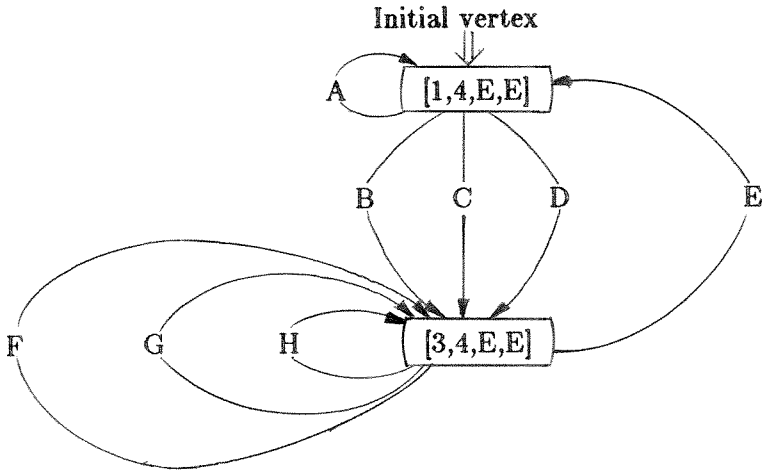


(b) A network for the proof of Theorem 2.

Figure 1. Network examples.



(a) Acyclic versions of M and N in Figure 1a with respect to {[1,4,E,E], [3,4,E,E]}

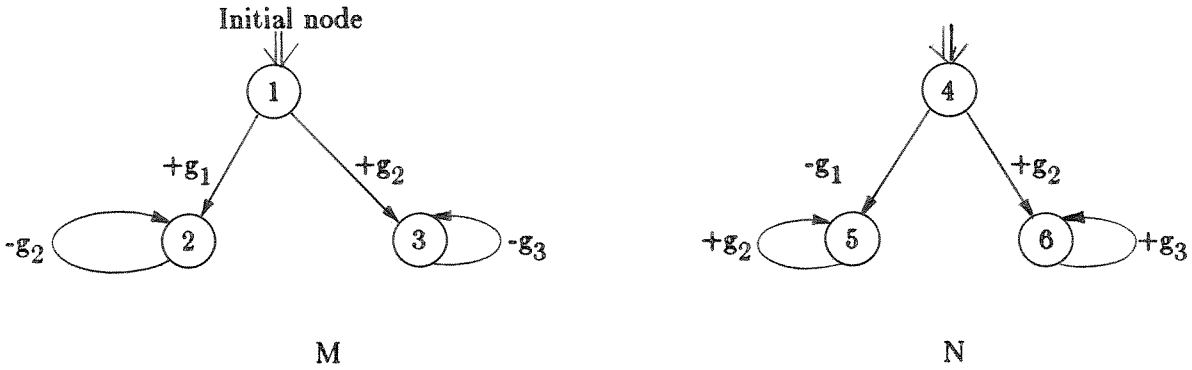


List of labels in the closed cover graph:

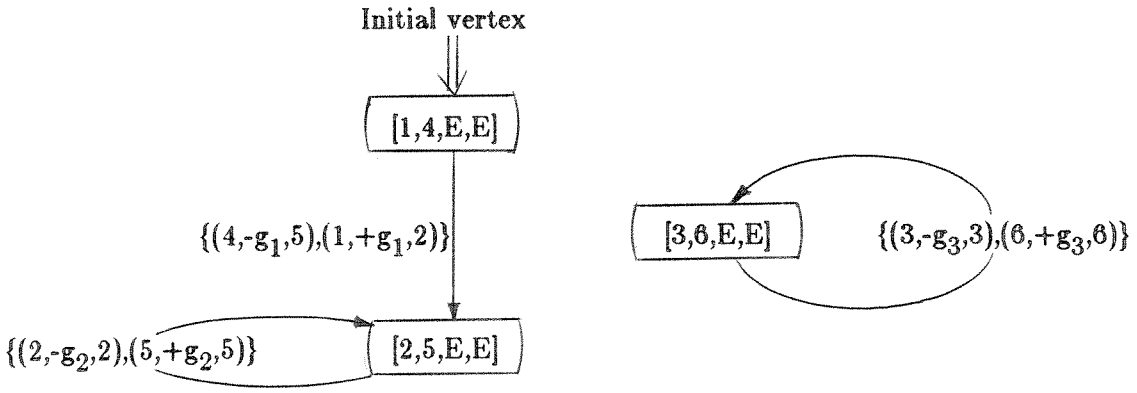
A={(1, -Ndata, 2),(4, +Ndata, 5),(5, -Ack, 4),(2, +Ack, 1)}
B={(1, -Err, 2),(4, +Err, 6),(6, -Err, 4),(2, +Err, 3)}
C={(1, -Err, 2),(4, +Err, 6),(6, -Nack, 4),(2, +Nack, 3)}
D={(1, -Ndata, 2),(4, +Ndata, 5),(5, -Err, 4),(2, +Err, 3)}
E={(3, -Ldata, 2),(4, +Ldata, 5),(5, -Ack, 4),(2, +Ack, 1)}
F={(3, -Ldata, 2),(4, +Ldata, 5),(5, -Err, 4),(2, +Err, 3)}
G={(3, -Err, 2),(4, +Err, 6),(6, -Nack, 4),(2, +Nack, 3)}
H={(3, -Err, 2),(4, +Err, 6),(6, -Err, 4),(2, +Err, 3)}

(b) Closed cover graph for the closed cover {[1,4,E,E], [3,4,E,E]}

Figure 2. Example 1.

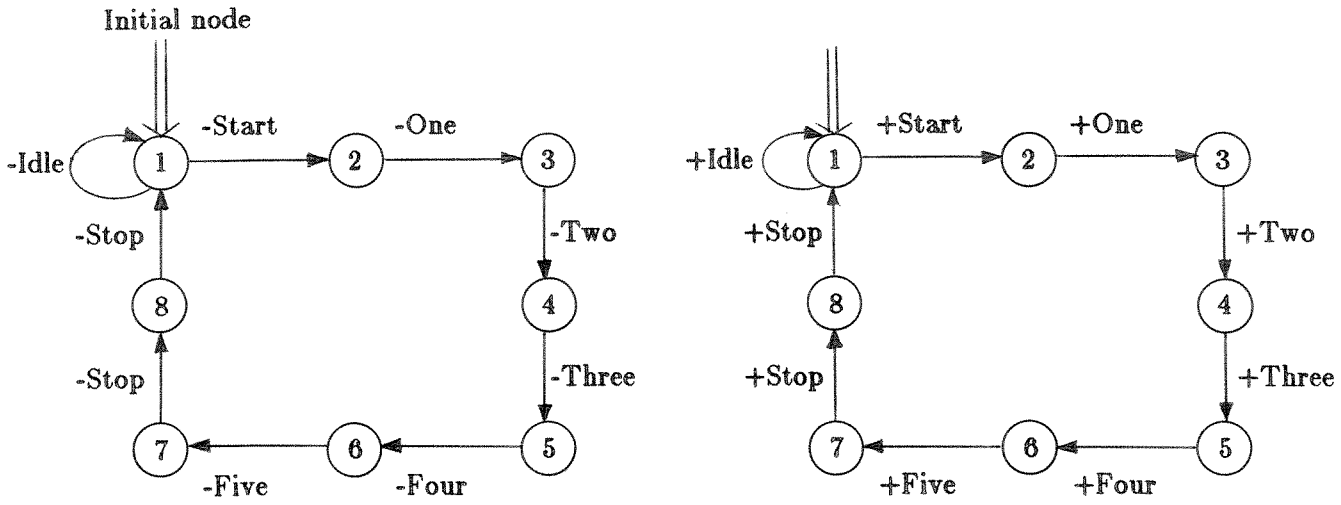


(a) Network(M,N)

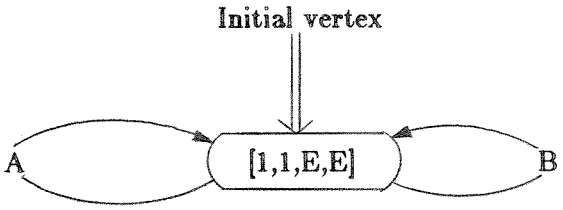


(b) The closed cover graph of the closed cover : C={[1,4,E,E], [2,5,E,E], [3,6,E,E]} for (M,N)

Figure 3. A network for the proof of Lemma 5.



(a) Network (M,N)



List of shown labels:

A={(1,-Idle,1),(1,+Idle,1)}
B={(1,-Start,2),(1,+Start,2),(2,-One,3),(2,+One,3),
(3,-Two,4),(3,+Two,4),(4,-Three,5),(4,+Three,5),
(5,-Four,6),(5,+Four,6),(6,-Five,7),(6,+Five,7),
(7,-Stop,8),(7,+Stop,8),(8,-Stop,1),(8,+Stop,1)}

(b) A closed cover graph for network (M,N)

Figure 4. A start-stop protocol example.

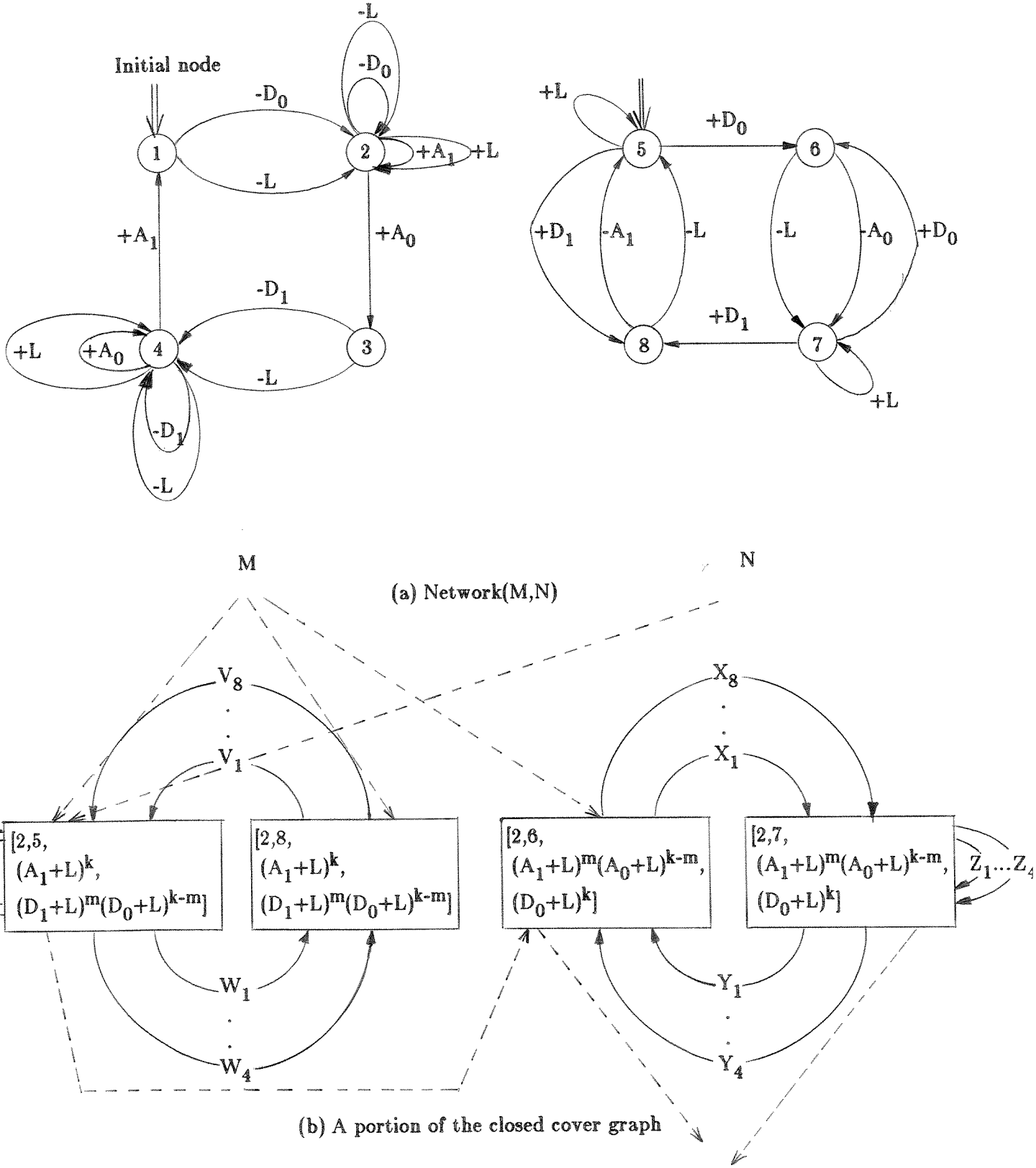


Figure 5. An Alternating Bit protocol example.

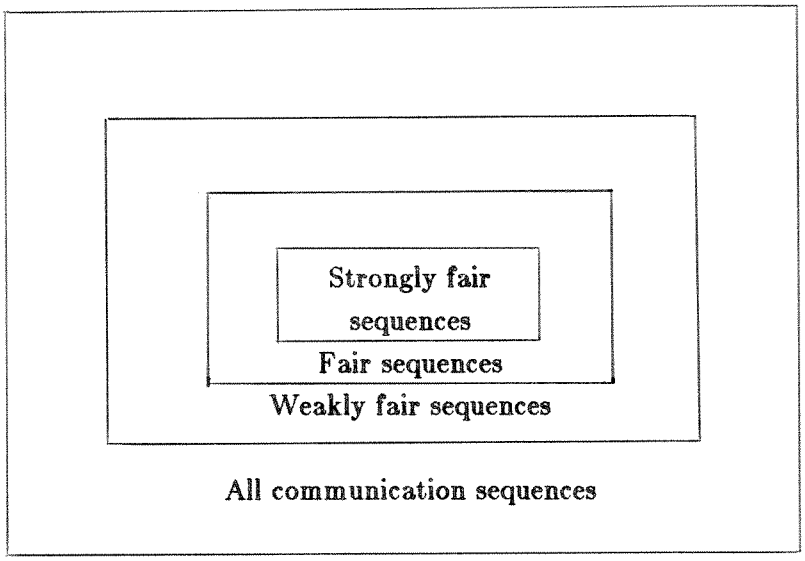
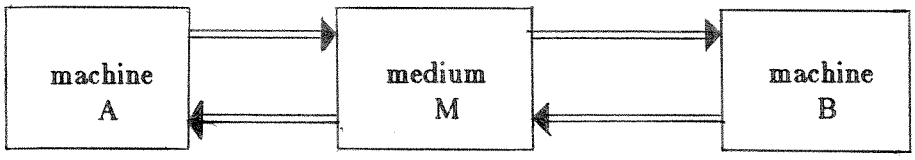
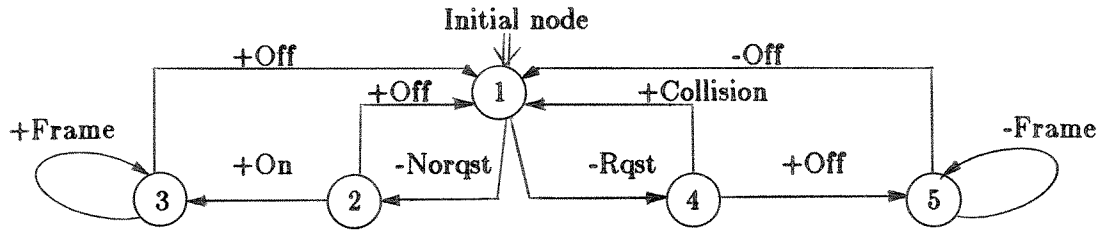


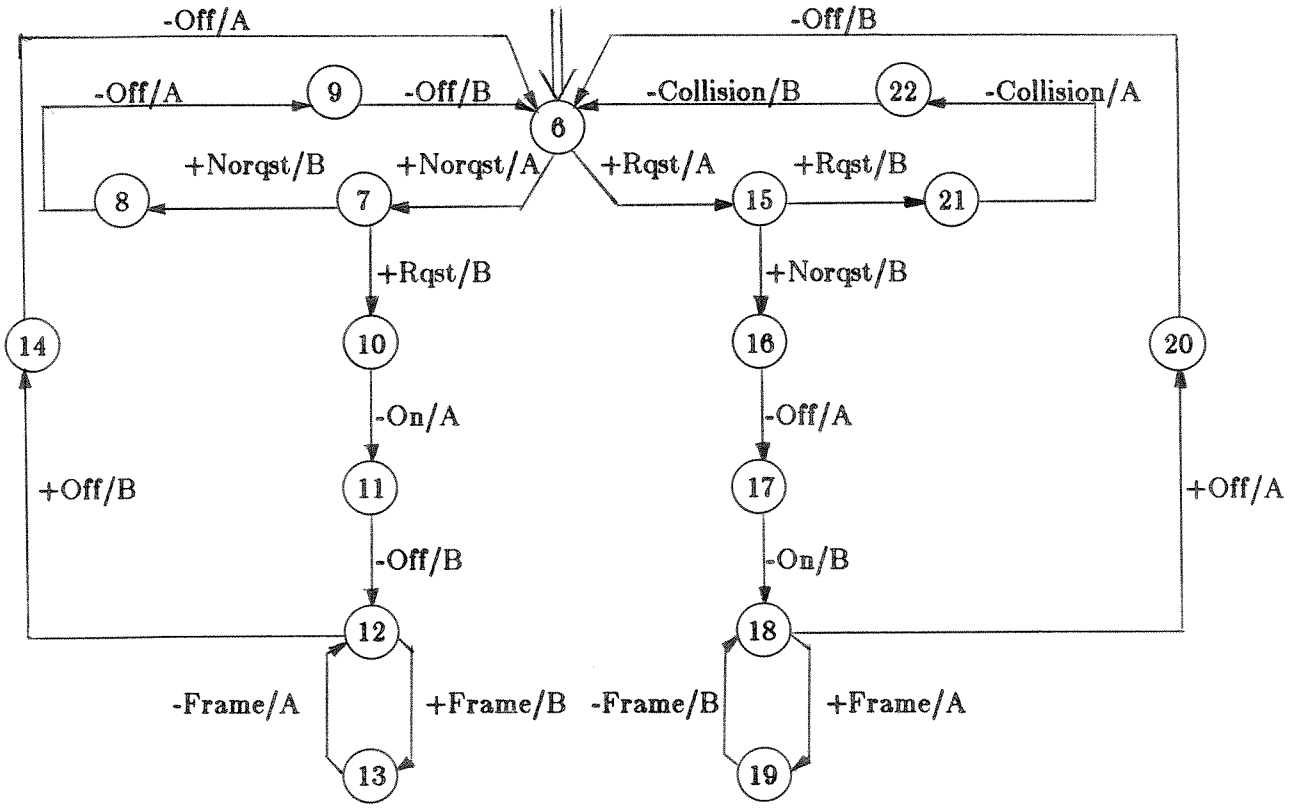
Figure 7. Relationships between the different classes of communication sequences.



(a) the network



(b) machine A or B



(c) machine M

Figure 6. A CSMA/CD protocol example.