AN EXAMPLE FOR CONSTRUCTING COMMUNICATING MACHINES BY STEP-WISE REFINEMENT

Mohamed G. Gouda

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

TR-223 March 1983

ABSTRACT

We consider the problem of constructing two finite-state machines that communicate by exchanging messages via two, one-directional, unbounded, FIFO channels. The two machines should be constructed such that their communication is guaranteed to progress indefinitely. We discuss a methodology to solve this problem by a succession of refinement steps. At each step more nodes and edges are added to the two machines constructed so far; this continues until the required two machines are realized. We illustrate the usefulness of this methodology by using it to construct two communicating machines which model the call establishment/clear protocol in X.25.

I. INTRODUCTION

Many communication protocols, notably the Alternating-bit protocol [1], and the call establishment/clear protocols in X.21 [5], X.25 [1], and X.75 [7], can be modeled as two finite-state machines which communicate over two one-directional, unbounded, FIFO channels [6]. In each of these models, the communication between the two machines are required to progress indefinitely. In this paper, we propose a new methodology for constructing a pair of such machines whose communication is guaranteed to progress indefinitely. The methodology is based on the well known principle of step-wise refinement [2].

The paper is organized as follows. The model of communicating machines is discussed in Section II. Then in Section III, we outline a step-wise refinement methodology for constructing a pair of communicating machines whose communication is guaranteed to progress indefinitely. An example of using this methodology to construct a machine pair that models the call establishment/clear protocol in X.25 is discussed in Section IV. Concluding remarks are in Section V.

II. COMMUNICATING MACHINES

A communicating machine M is a directed labelled graph with two types of edges, namely sending and receiving edges. A sending (or receiving) edge is labelled send(g) (or receive(g), respectively) for some message g in a finite set G of messages. Each node in M has at least one output edge. One of the nodes in M is identified as its initial node; each node is reachable by a directed path from the initial node. A node in M whose output edges are all sending (or receiving) edges is called a sending (or receiving, respectively) node; otherwise it is called a mixed node.

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

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

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

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

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

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

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

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

A state s of M and N is said to be reachable iff it is reachable from the initial state of M and N.

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

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

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

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

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

We say that the communication between two communicating machines will progress indefinitely iff the communication is free from deadlocks and unspecified receptions. In this paper, we are interested in constructing pairs of communicating machines whose communications are guaranteed to progress indefinitely.

III. STEP-WISE REFINEMENT

Assume that it is required to construct two communicating machines M and N such that their communication is guaranteed to progress indefinitely. And assume that the required M and N are expected to be complex; i.e., they are expected to contain large numbers of nodes and edges. In this case, the task of constructing M and N is better divided into a succession of refinement steps. At the ith step, i=1,2,...,k (for some k), two communicating machines M_{i+1} and N_{i+1} are constructed by adding more nodes and edges to the two communicating machines M_i and N_i, constructed earlier. These additions should be such that if the communication between M_i and N_i is guaranteed to progress indefinitely, then also the communication between M_{i+1} and N_{i+1} is guaranteed to progress indefinitely. M_1 and N_1 are called the initial machines; their communication should be guaranteed to progress indefinitely. M_{k+1} and N_{k+1} are called the final machines; they are the required machines M and N; and their communication is guaranteed, as required, to progress indefinitely.

To make this methodology useful, one needs a technique to add nodes and edges to two communicating machines while preserving their freedom of communication deadlocks and unspecified receptions. Such a technique is discussed next.

Let M_i and N_i be two communicating machines where it is required to add more nodes and edges. Then, the next four steps can be followed:

i. Prove that the communication between M_i and N_i will progress indefinitely. (The proof is only necessary if M_i and N_i are the initial machines; if M_i and N_i are the product of a previous refinement step, then their communication is already guaranteed to be free of deadlocks and unspecified receptions.)

ii. Find two compatible nodes v_i and w_i, in M_i and N_i respectively, that satisfy some conditions (discussed later).

iii. Construct two communicating submachines m_i and n_i that satisfy some conditions (discussed later). (A communicating submachine is a communicating machine where some nodes are designated terminal nodes. Also, a node in a submachine may have no output edges.)

iv. Replace node v_i in M_i by submachine m_i and node w_i in N_i by submachine n_i according to some rules (discussed later). The result of this substitution is two communicating machines M_{i+1} and N_{i+1} whose communication is guaranteed to progress indefinitely.

These four steps, which constitute a single refinement step, are discussed in detail in [ ], along with a correctness proof to show that the communication of the resulting machines M_{i+1} and N_{i+1} will indeed progress indefinitely. In the next section, we use these steps to construct a call establishment/clear protocol, for X.25, whose communication is guaranteed to progress indefinitely.

IV. AN EXAMPLE

Assume that it is required to construct two communicating machines M and N which exchange the messages g_1 to g_7. (Although irrelevant to the following discussion, these messages stand for the following:

g_1 stands for call request.
g_2 stands for call connected.
g_3 stands for incoming call.
g_4 stands for call accepted.
g_5 stands for clear request.
g_6 stands for clear indication.
g_7 stands for clear confirmation.)

Also assume that the designer starts with the two initial machines M_1 and N_1 shown in Figures 1a and 1b respectively. (For convenience, edge labels such as send(g) and receive(g) are written as -g and +g, respectively.) Following the above refinement procedure, the designer should carry out the next steps.

First Step:

Since the given machines are initial machines, the designer should first prove that their communication will progress indefinitely. Such a proof is not straightforward, even though the two machines have a small number of nodes and edges. This is because the communication between M_1 and N_1 is unbounded (i.e., there is an infinite number of reachable states); hence, conventional state exploration [8] cannot be used to prove freedom of deadlocks and unspecified receptions.

Fortunately, a recent technique [3] can be used in this case, to prove indefinite progress. This technique is based on constructing a set C of states of M_1 and N_1, then proving that C satisfies three conditions:

i. The initial state of M_1 and N_1 must be in C.

ii. Each directed cycle in M_1 or N_1 must have at least one node referenced in some state in C.

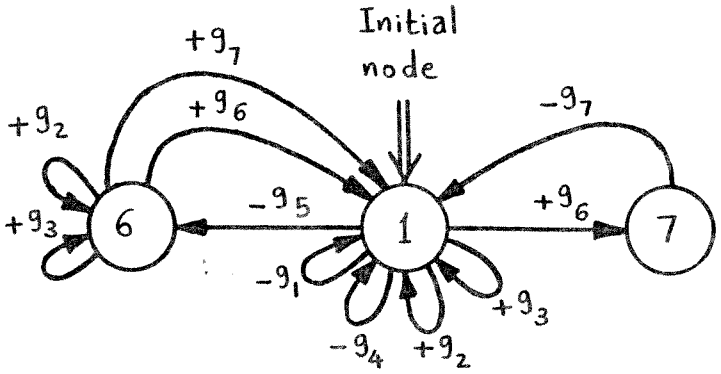
iii. The acyclic version of M_1 with respect to C can be constructed from M_1 by partitioning each node v, which is referenced in some state in C, into two nodes. One node has all the input edges of v and no output edges; the other node has all the output edges of v and no input edges. Similarly, the acyclic version of N_1 with respect to C can be defined. The third condition can now be defined in terms of these acyclic versions. If the acyclic versions of M_1 and N_1 with respect to C start at a state s_1 in C and if they reach a state s_2 after which no other state is reachable, then state s_2 must also be in C.

It is shown in [3] that the existence of such a set C, called a closed cover for M_1 and N_1, guarantees that the communication between M_1 and N_1 will progress indefinitely.

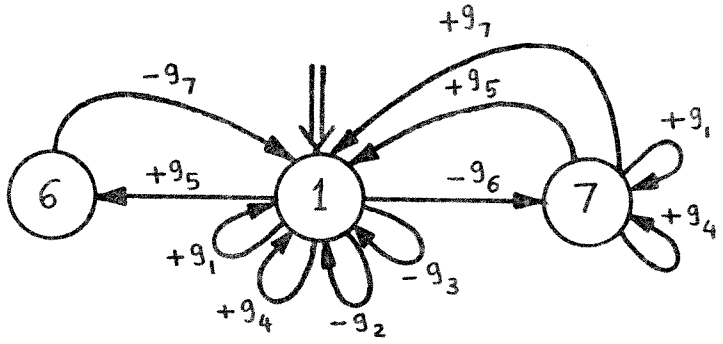
For the two machines M_1 and N_1 in Figures 1a and 1b, consider the following set:

$$
\begin{aligned}
C = \{\ & [1,1,(g_2+g_3)^n,\ (g_1+g_4)^n], \\
& [6,6,E,E], \\
& [7,7,E,E], \\
& [6,7,(g_2+g_3)^n\ g_6,\ (g_1+g_4)^n\ g_5], \\
& [6,1,(g_2+g_3)^n,\ (g_1+g_4)^{n-1}\ g_5], \\
& [1,7,(g_2+g_3)^{n-1}\ g_6,(g_1+g_4)^n]\}
\end{aligned}
$$

where + is the usual "or" operator, and E denotes the empty string. Each element in C denotes a set of states of M_1 and N_1; for example $[1,1,(g_2+g_3)^n, (g_1+g_4)^n]$ denotes the set:

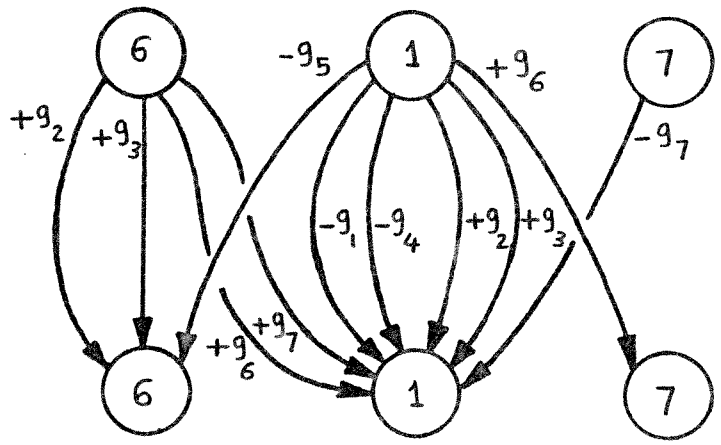


(a) M_1

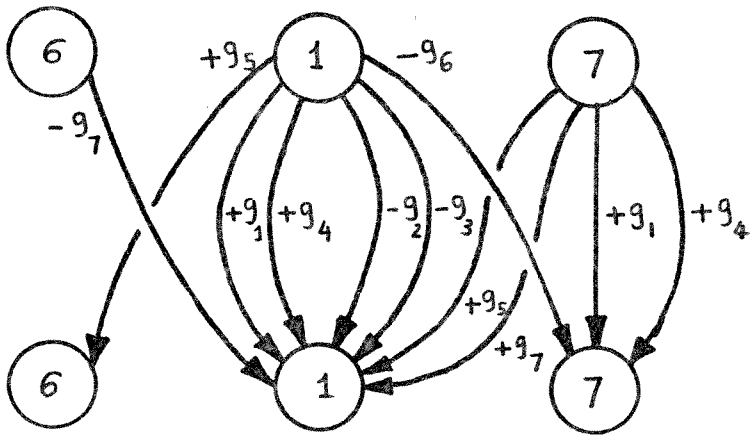


(b) N_1

Fig. 1 Initial Machines



(a) A_1



(b) B_1

Fig. 2 Acyclic versions of the initial machines with respect to closed cover C.

{ [1,1,E,E],
[1,1,g_2,g_1], [1,1,g_2,g_4], [1,1,g_3,g_1], [1,1,g_3,g_4],
[1,1,g_2 g_3, g_1 g_4],...
[1,1,g_3 g_3 g_3, g_4 g_1 g_4],
...}.

Similarly, the element [6,6,E,E] in C denotes the singleton {[6,6,E,E]}.

A state s of M_1 and N_1 is said to be in C iff s is in the set denoted by an element of C. Next, we show that C satisfies the above three conditions of a closed cover:

i. The initial state [1,1,E,E] is in C since it is in the element [1,1,$(g_2+g_3)^n$, $(g_1+g_4)^n$] of C.

ii. Each node in M_1 or N_1 is referenced by at least one state in C. For instance, nodes 1 and 1 are referenced by the state [1,1,E,E] in C; nodes 6 and 6 are referenced by the state [6,6,E,E] in C; nodes 7 and 7 are referenced by the state [7,7,E,E] in C.

iii. Since each node in M_1 or N_1 is referenced by at least one state in C, the acyclic versions of M_1 and N_1 with respect to C are as shown in Figures 2a and 2b; they are called A_1 and B_1 respectively. It can be shown, by enumeration, that if A_1 and B_1 start at any state in C, and if they reach a state s after which no other state is reachable, then state s must also be in C. For example, assume that A_1 and B_1 start at a state in [1,1,$(g_2+g_3)^n$, $(g_1+g_4)^n$]. If A_1 sends g_1 or g_4 and B_1 receives g_1 or g_4, then the resulting state must be in [1,1,$(g_2+g_3)^n$, $(g_1+g_4)^n$]; i.e., it is in C. If A_1 sends g_5 and B_1 receives g_1, then the resulting state must be in [6,1,$(g_2+g_3)^n$, $(g_1+g_4)^{n-1}$ g_5]; i.e., it is in C. If A_1 sends g_5 and B_1 sends g_8, then the resulting state must be in [6,7,$(g_2+g_3)^n$ g_6, $(g_1+g_4)^n$ g_5], and so in C. Similarly, assume that A_1 and B_1 start at a state in [6,7,$(g_2+g_3)^n$ g_6, $(g_1+g_4)^n$ g_5]. If A_1 receives g_6, then n=0, B_1 must receive g_5, and the resulting state must be [1,1,E,E] which is in C, and so on.

Therefore, set C is a closed cover for the initial machines M_1 and N_1; hence the communication between M_1 and N_1 is guaranteed to progress indefinitely.

Second Step:

The designer should find two compatible nodes v_1 and w_1, in machines M_1 and N_1 respectively, which satisfy the following two conditions:

i. Node v_1 has an output edge labelled send(g) (or receive(g)) iff node w_1 has an output edge labelled send(g) (or receive(g), respectively).

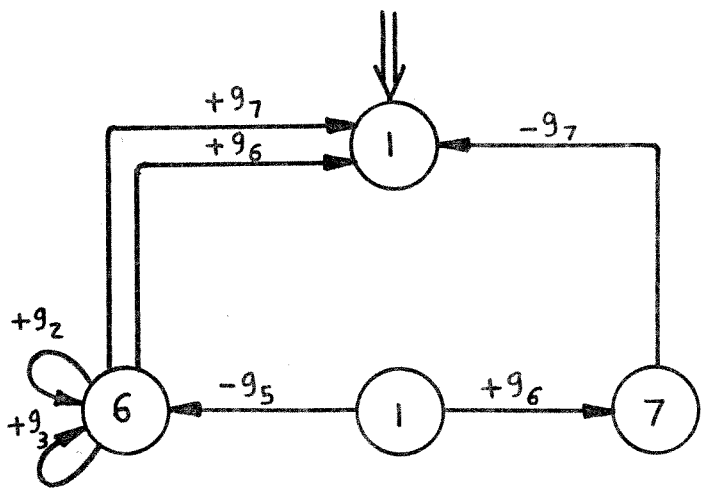
ii. The second condition should be satisfied after removing the self-loops at nodes v_1 and w_1. Informally, if machine M_1 reaches node v_1 for the jth time and if at the same instant machine N_1 reaches node w_1 for the jth time (j=1,2,...), then the two channels between M_1 and N_1 must be empty at this same instant.

Next, we show that nodes 1 and 1 in M_1 and N_1 satisfy these two conditions:

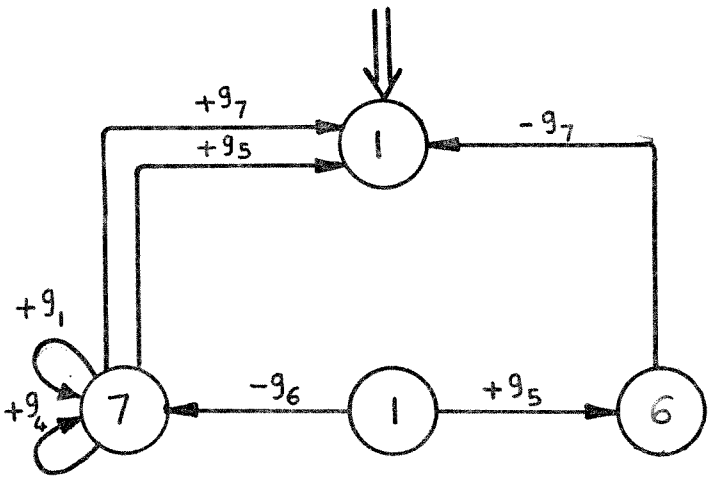
i. Obviously, nodes 1 and 1 satisfy condition i.

ii. To show that nodes 1 and 1 satisfy condition ii, for j=1,2,..., we use an induction argument.

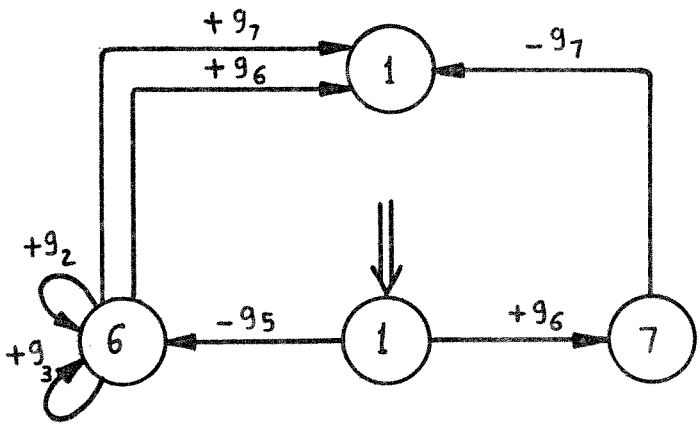
For j=1: Remove the self-loops at nodes 1 and 1 in M_1 and N_1. Then, partition each node 1 into two nodes; one has all the input edges of the original node; and the other has all the output edges of the original node. The resulting machines are C_1 and D_1, shown in Figures 3a



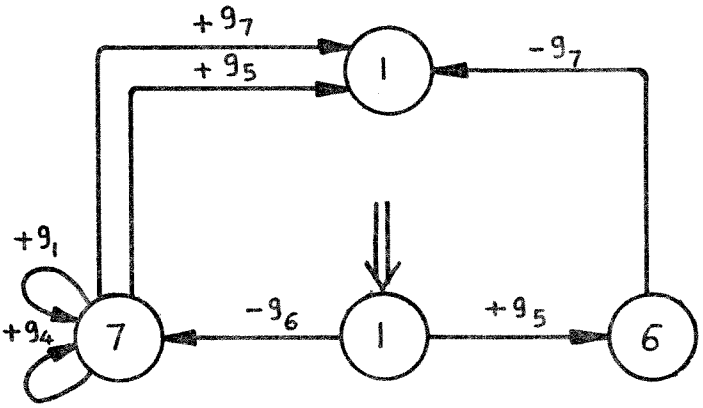
(a) C_1



(b) D_1



(c) E_1



(d) F_1

Figure 3 Proving that nodes 1 and 1 in M_1 and N_1 are compatible

and 3b. Since the only reachable state of C_1 and D_1 is [1,1,E,E], then nodes 1 and 1 satisfy condition ii for j=1.

If condition ii is satisfied for j=k, then it is satisfied for j=k+1: Remove the self-loops at nodes 1 and 1 in M_1 and N_1; then partition each node 1 as before. In each machine, make the partitioned node, with the output edges, the initial node of the machine. The resulting machines are E_1 and F_1, shown in Figures 3c and 3d. It is straightforward to show, by conventional state exploration, that if E_1 and F_1 reach a state [1,w,x,y] or [v,1,w,y], then the two machines must eventually reach [1,1,E,E]. Therefore, if nodes 1 and 1 satisfy condition ii for j=k, then they satisfy it for j=k+1.

This completes the proof that nodes 1 and 1 in machines M_1 and N_1 are compatible.

Third Step:

The designer should then construct two communicating submachines m_1 and n_1 which satisfy the following two conditions:

i. Let v_1 and w_1 be the two compatible nodes, in M_1 and N_1 respectively, selected in the previous step. Submachine m_1 has an edge labelled send(g) (or receive (g)), only if node v_1 has a self-loop labelled send(g) (or receive (g), respectively). Similarly, submachine n_1 has an edge labelled send(g) (or receive(g)), only if w_1 has a self-loop labelled send(g) (or receive(g), respectively).

ii. Each node in m_1 or n_1 is a terminal node. (Although this is a reasonable condition for the current example, it is a severe condition in general. In [4], we discuss how to relax this condition.)

Figures 4a and 4b show the two constructed submachines m_1 and n_1 for our example. Each node in m_1 or n_1 is designated a terminal node; and so m_1 and n_1 satisfy conditions i and ii.

Fourth Step:

Now, the designer should replace node 1 in machine M_1 (Figure 1a) by submachine m_1 (Figure 4a) as follows:

i. Remove all the self-loops at node 1.

ii. All the input edges of node 1 become input edges to the initial node of m_1.

iii. Each output edge of node 1 is replaced by an output edge (with the same label, and to the same destination node) at each terminal node in m_1. (There are other variations to this replacement step [4].)

The result is the final machine M in Figure 5a.

Similarly, node 1 in machine N_1 (Figure 1b) should be replaced by submachine n_1 (Figure 4b). The result is the final machine N in Figure 5b. Notice that the resulting two communicating machines M and N represent the call establishment/clear protocol in X.25 as defined in [1].

As shown in [4], the communication between the two final machines is guaranteed to progress indefinitely; i.e., the communication is free of deadlocks and unspecified receptions.

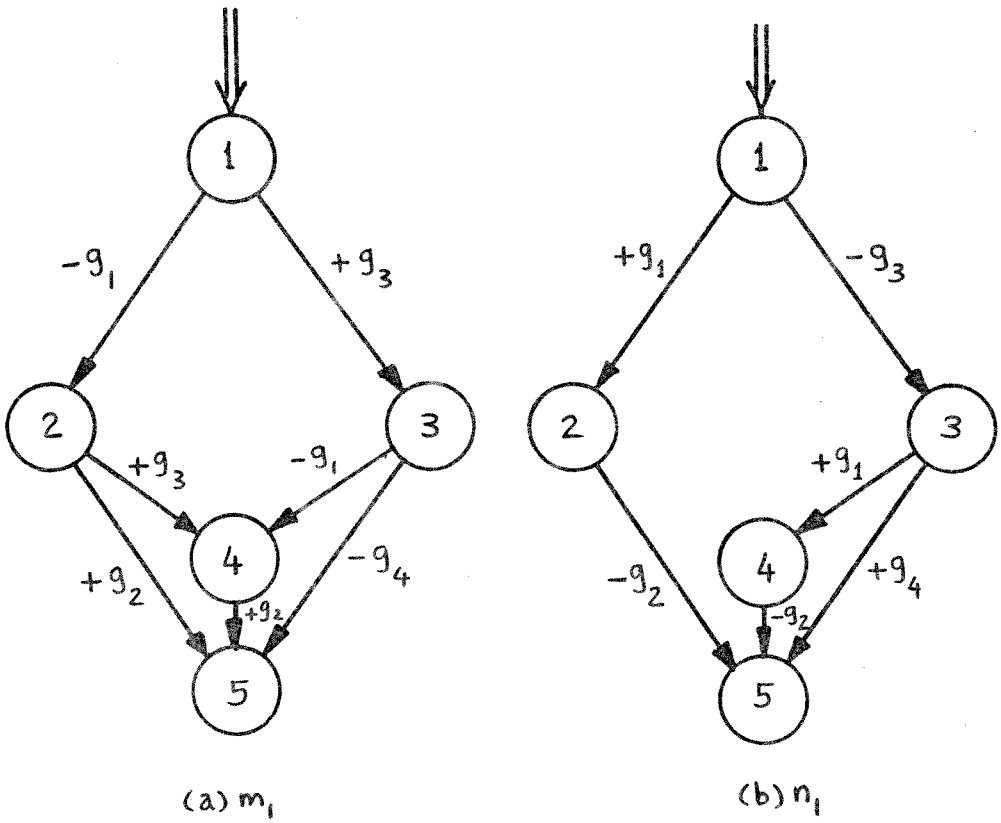
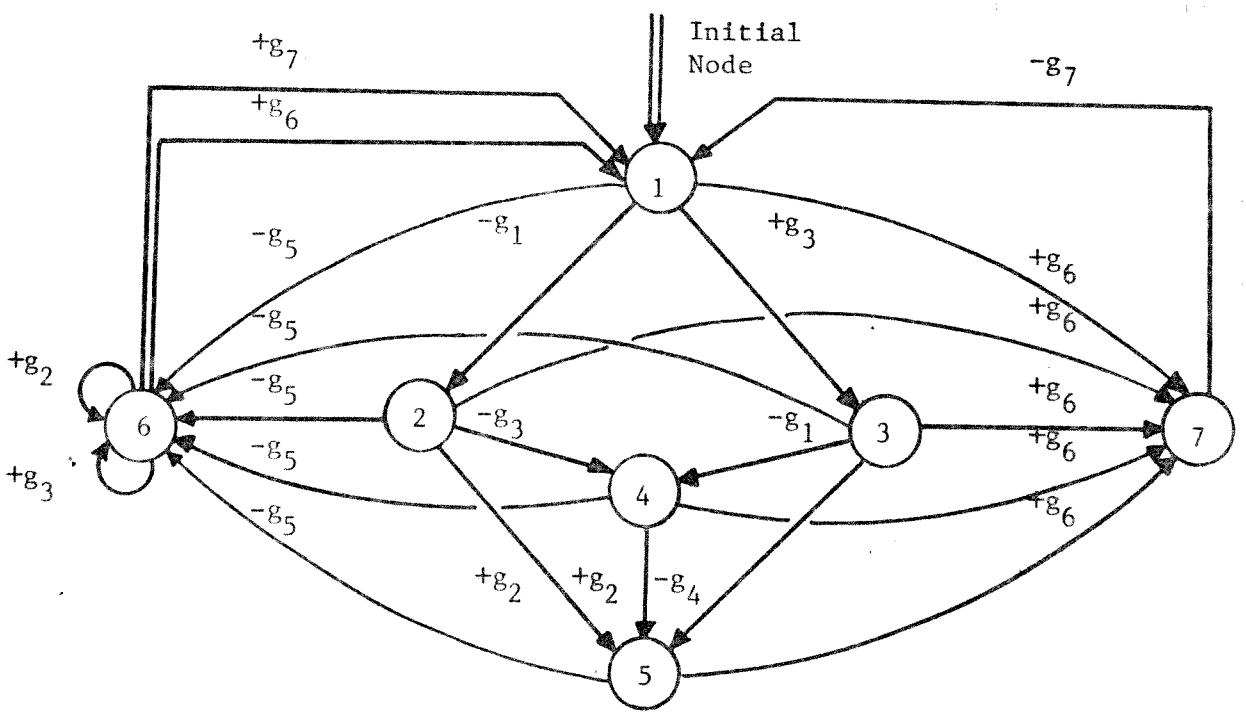
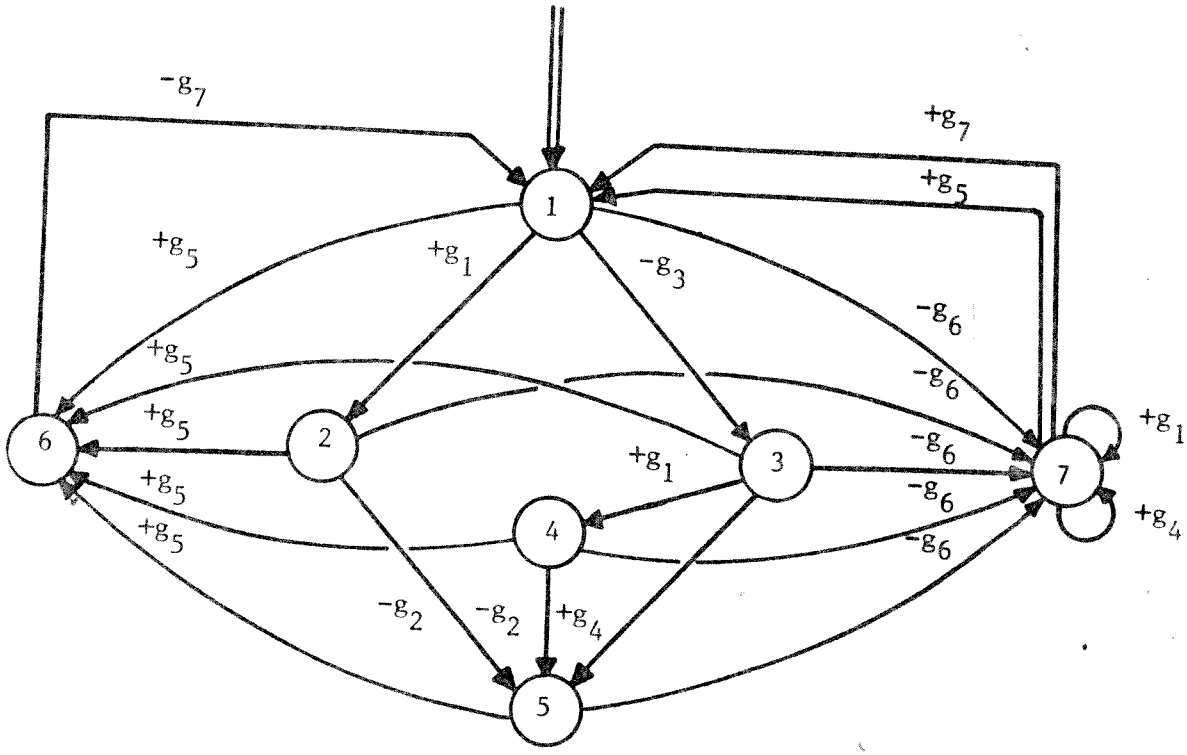


Figure 4 Two communicating submachines m_1 and n_1.



(a) M



(b) N

Figure 5 The call establishment/clear protocol in X.25.

V. CONCLUSIONS

We have demonstrated, by the virtue of one example, that step-wise refinement can be used to construct communicating machines that represent actual protocols. There are three advantages to using this methodology in constructing communicating machines:

i. Dealing with Complexity: This methodology allows one to construct very complex communicating machines via a succession of refinement steps. At each step, the designer adds more complexity to the communicating machines constructed so far. This continues until the required communicating machines with all their details are fully constructed.

ii. Computer Aided Design: Many of the steps in this methodology (e.g., checking a given closed cover, or verifying the compatibility of two given nodes) can be carried out by a computer. This suggests some computer aided design tools which can help the designer to concentrate on the creative aspects of the methodology.

iii. Increase our Understanding of the Constructed Machines: In the above example, the initial machines M_1 and N_1 and the two communicating submachines m_1 and n_1 were not selected arbitrarily to construct the required machines M and N. A closer examination of the example should reveal that m_1 and n_1 perform the call establishment function, while M_1 and N_1 perform the call clear function. (Hence the final machines M and N perform the call establishment/clear function as required.) In other words, this methodology encourages one to reason about each function of the protocol separately, before adding these functions together. This should lead to a better understanding of the protocol and its multi-functions.

ACKNOWLEDGEMENT:

The author is thankful to Prof. Bochmann for directing his attention to step-wise refinement. He is also thankful to K. F. Carbone for her speedy and careful typing.

REFERENCES

1. G. V. Bochmann, "Finite state description of communication protocols," Computer Networks, Vol. 2, Oct. 1978.

2. G. V. Bochmann, and M. Raynal, "Structured specification of communicating systems," IEEE Trans. on Computers, Vol. C-32, No. 2, Feb. 1983, pp. 120-133.

3. M. G. Gouda, "Closed covers: to verify progress for communicating finite state machines," Technical Report 191, Dept. of Computer Sciences, University of Texas at Austin, Jan. 1982, revised Jan. 1983.

4. M. G. Gouda, "Construction of communicating finite state machines by step-wise refinement," in preparation.

5. H. Rudin, C. H. West, and P. Zafiropulo, "Automated protocol validation," Computer Networks, Vol. 2, No. 415, 1978, pp. 373-380.

6. C. A. Sunshine, "Formal modeling of communication protocols," USC/Information Science Institute, Research Report 81-89, March 1981.

7. S. T. Voung, and D. D. Cowan, "Automated protocol validation via resynthesis: The CCITT X.75 packet level recommendation as an example," Tech. Report CS-80-39, Dept. of Comp. Science, Univ. of Waterloo, revised May 1981.

8. C. H. West, "An automated technique of communication protocol validation," IEEE Trans. on Commun., Vol. COM-26, pp. 1271-1275, Aug. 1978.