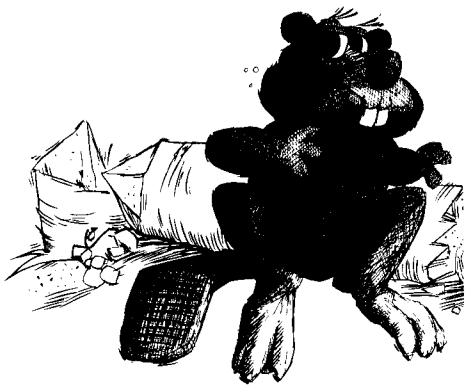
UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT

UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT

UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT



Automated Protocol Validation Via Resynthesis: The CCITT X.75 Packet Level Recommendation As An Example

Son T. Vuong
Donald D. Cowan

CS-80-39

August, 1980
Revised May, 1981

AUTOMATED PROTOCOL VALIDATION VIA RESYNTHESIS: THE CCITT X.75 PACKET LEVEL RECOMMENDATION AS AN EXAMPLE

Son T. Vuong and Donald D. Cowan

Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1

ABSTRACT

Using an automated protocol synthesis system which can handle certain types of potential design errors such as unspecified receptions, "dead codes", state deadlocks, and state ambiguities, we validate a subset of the packet level of the X.75 protocol recommended by the International Telegraph and Telephone Consulative Committee (CCITT). The problem of the uniqueness of a single, composite state diagram for the STEs is discussed. The validation procedure identifies a number of points where the state diagram specification does not completely define the behavior of the X.75 packet level protocol.

AUTOMATED PROTOCOL VALIDATION VIA RESYNTHESIS: THE CCITT X.75 PACKET LEVEL RECOMMENDATION AS AN EXAMPLE

Son T. Vuong and Donald D. Cowan

Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1

I. INTRODUCTION

The increasing complexity of communication protocols for computer networks and distributed systems in recent years has intensified the demand for computer tools for protocol synthesis and validation. One such tool, a synthesizer similar to the one developed by Zafiropulo *et al.* [1], allows the designer to construct his protocol in a stepwise interactive manner such that the synthesized protocol is free from certain types of errors such as unspecified receptions, "dead codes", state deadlocks, and state ambiguities. This synthesis tool which is based on a set of production rules, can also be used for protocol validation. Indeed, from a given protocol specification, we can apply the synthesis procedure to construct an error-free copy of the given protocol, and thus carry out the validation by simply comparing the original protocol specification with the synthesized one.

In this paper, the validation system using the production rules is employed in a real environment to validate a subset of the packet level of X.75 as an example of a reasonably complex protocol. The X.75 protocol has been chosen because it is a standard protocol for network interconnection, an area which is currently of widespread interest, and because it is formally specified in state diagram form and so it can be readily validated.

We first describe the X.75 protocol and how we interpret its specification for the purpose of validation. Then we outline the validation procedure and a decomposition method which facilitates the validation process. Finally the validation results are presented and their significance discussed.

II. X.75 DESCRIPTION

The CCITT Recommendation X.75 [2] is a standard protocol which provides virtual circuit service for the interconnection of public data networks. The use of X.75 is illustrated in Figure 1, where a virtual circuit between two DCEs connected to two different packet switching networks (PSNs) is formed by the concatenation of an X.75, two X.25 and two internal virtual circuits. The term DTE (Data Terminal Equipment) is used for customer equipment (host, terminal) using the public network, DCE (Data Circuit terminating Equipment) for access to the source or destination network switching node, and STE (Signalling TErminal) for network interconnection node (sometimes called gateway half).

The X.75 protocol is in principle based on Recommendation X.25 suitably adapted to meet the requirements of a terminal and transit interexchange call control system. The basic system

structure consists of three levels, the physical, link and packet levels:

i) the physical level specifies the mechanical, electrical, functional, and procedural interface characteristics between the transmission media and the signaling terminals (STEs) of the two networks, and so, it provides a bit-serial, full-duplex, point-to-point circuit for digital transmission;

ii) the link level consists of the packet transfer procedures which operate over the error-prone physical circuits and provide a mechanism for reliable transport of packets between the two STEs; and

iii) the packet level specifies the packet signalling procedures for the exchange of control and data packets between the two STEs. This level provides facilities for establishing, maintaining, and terminating virtual circuits. Multiplexing of several virtual circuits onto a single data link, and flow control are also functions of the packet level.

We have only applied the validation procedure to the X.75 packet level protocol without the Restart and Flow Control procedures. In fact, the term "X.75" is often used in this paper to mean just this major subset of the X.75 packet level protocol.

From the X.75 document, X.75 consists of a number of procedures which are described in three ways:

1. A text description of the states of the channel between the two STEs (Section 3 of [2]).
2. State diagrams specifying the packets sent between the STEs and the possible state sequences (Annex 2 of [2]). The state diagrams describing the Call Establishment, Clearing, Reset, and Interrupt Transfer procedures are reproduced as shown in Figure 2.
3. Action tables describing the action, if any, to be taken by an STE on receipt of any kind of packet or upon a certain timeout, and specifying the state the STE enters following the action (Annexes 3 and 4 of [2]).

Because state diagrams generally provide a more precise protocol description than prose, and as clearly stated in [2], "Annex 2 defines the states of the X/Y interface and the transitions between states in the normal case", we interpret the state diagrams as being the definitive specification of X.75 and the text description and action tables as providing supplementary information to help understand the state diagrams. In a later section, after the normal case has been validated, we present the discrepancies among the text description, the action table description and the state diagram specification, and discuss their implications.

With this interpretation, our next task was to derive the logical structure of the STE-X and STE-Y processes which were to communicate according to the combined state diagrams shown in Figure 2. This task was necessary because the validation procedure can only be applied to a pair of separately defined processes. Besides, the protocol specification must be derived in terms of a pair of local state diagrams in order to implement the protocol.

Whereas the provision of the combined state diagram specification in the X.75 document implies that one can derive a unique pair of state diagrams from a combined state diagram, an algorithm for such derivations is not given in the X.75 document. We are thus led to investigate the existence and construction of such an algorithm.

III. X.75 SPECIFICATION IN TERMS OF LOCAL STATE DIAGRAMS

Figure 2a explains the notation used in the X.75 combined state diagram specification. All transitions in a combined state diagram represent either an STE-X or STE-Y initiated transition. This is indicated by a label (STE-X or STE-Y) on the arc between each pair of states. Another label on the arc represents an event type transmitted by the specified STE and subsequently received by the other STE. This notation apparently implies that each transition in a combined state diagram is associated with an information exchange between the two processes STE-X and STE-Y. As such, it can be modeled by a related pair of send/receive transitions in the separate local state diagrams of STE-X and STE-Y as illustrated in the example of Figure 3a. In this example, the transition labeled "STE-X" and "Call request" from the Ready state (p1) to the 'STE-X Call Request' state (p2) is transformed into a pair of send/receive transitions, one in STE-X and one in STE-Y as follows:

i) in STE-X, the transition is from state 1 to state 2 to transmit a Call request packet (denoted by the negative label -CAR); and

ii) in STE-Y, the transition corresponds to the reception of the Call request packet (denoted by the positive label +CAR) to go also from state 1 to state 2.

By repeatedly applying this transformation to each transition of a combined state diagram, the corresponding local state diagrams for STE-X and STE-Y can be simply derived. This derivation was introduced by West and Zafiropulo [3] with regard to the X.21 state diagrams.

The state diagrams for STE-X and STE-Y derived this way obviously have the same topology (as the original combined state diagram) and the "send/receive pairwise" property, i.e. there is a one-to-one correspondence between a transition transmitting an event in one process and a transition receiving the same event in the other process. We call such a protocol a "send/receive symmetric" protocol.

Using a deterministic algorithm such as the one described above to derive the logical structure of the two processes, the combined state diagram specification can only model a restricted class of interactions, namely the symmetric and half-duplex interactions (half-duplex interactions ensure the "send/receive pairwise" property). In full-duplex interactions, however, message collisions are possible (two messages are said to collide when neither is received before the other is transmitted). Thus, to model symmetric full-duplex interactions, a transition in a combined state diagram would have to be interpreted either as a pair of send/receive transitions (Figure 3a), or as a single reception transition (Figure 3b) resulting from a collision. The former interpretation is, clearly, the normal case. The latter interpretation can only be adopted under the following conditions:

i) the transmission transition cannot possibly be specified owing to some semantic consideration; and

ii) the reception transition occurs because of a collision.

We explain these latter conditions for each and every case in the X.75 specification where a combined state diagram transition has been simply interpreted as a single reception transition in a local state diagram.

In the Call Establishment procedure (Figure 2b), the transition from 'Call Request' state p2 (or p3) to the 'Call Collision' state (p5) is interpreted simply as a reception transition from state 2 to state 4 in STE-X (or from state 3 to state 4 in STE-Y), as shown in Figure 4b. We have the following justifications. First, the companion call request transmission arc from state 2 in STE-X (or state 3 in STE-Y) should not be specified because it would not make sense for an STE to

request for a call setup after it received a call request from the other STE; Indeed, it should only confirm the call request received by sending a Call connected packet. Second, the reception of a call request in state 2 in STE-X (or state 3 in STE-Y) can be specified as the result of the transmission of a call request in state 1 (Ready state) by STE-Y (or STE-X) in collision with the call request transmission by STE-X (or STE-Y).

Similarly, in the Call Clearing procedure (Figure 2d), the self-loop transition with label "Call Connected, Data, Interrupt, Flow Control, or Reset" from reception transition from state 11 in STE-X (or state 12 in STE-Y), as shown in Figure 4a. The companion self-loop transmission arc from state 12 in STE-Y (or state 11 in STE-X) is omitted as it is, obviously, meaningless to transmit a packet of any of the above types while in a Clear Request state. This packet can be received, however, as the result of the transmission of the same packet by the other process in collision with the clear request transmission by the former process.

Likewise, the self-loop transition with label "Data, Interrupt, or Flow Control" from a Reset Request state (d2 or d3) in the Call Reset procedure (Figure 2d) is also mapped into a self-loop reception from state 8 in the appropriate diagram, as shown in Figure 4c.

We have, thus, derived the local state diagram specification for the four procedures of X.75, shown in Figure 4 and simply combined these procedures to form the entire X.75 specification, shown in Figures 5a and 5b.

It should be mentioned that the effect of timeouts is not specified in the local state diagram specification of X.75. A transition which may take place after a timeout is simply treated as a normal transition which may occur as though there were no timeout. We have, therefore, not addressed the problem of validating the timeouts which are defined to ensure continuing operation of the protocol when either STE fails to respond to the other within a reasonable time. The timeout lengths generally depend on the operating environment and the implementations. An evaluation of the actions taken can only be made after a detailed study of the possible causes of a response failure. Such a study is beyond the scope of our current validation.

In deriving the specification of X.75, we have faced the same problem of state confusion and inadequate representation of the combined state diagram model, which has been discussed before by West and Zafiropulo [3] with regard to the X.21 protocol, and by Belnes and Lynning [4] and Bochmann [5] with respect to the X.25 packet level protocol. In the next section, we comment on the state confusion problem from a different point of view, and demonstrate that the current combined state diagram model can be slightly modified to result in a complete and concise model for defining symmetric protocols such as X.75.

IV. MODIFIED STATE DIAGRAMS AND STATE CONFUSION PROBLEM

Although by using a nondeterministic algorithm, we have derived the local state diagram specification of X.75, which, we believe, is correct, a nondeterministic algorithm is generally undesirable because every time the algorithm is applied to the same combined state diagram, we have no guarantee that a unique pair of local state diagrams will be produced. Consequently, it is conceivable that equipment designed by different manufacturers to implement the same protocol standard may not be compatible. A desirable alternative is, therefore, to modify the combined state diagram specification such that once a combined state diagram is given, the desired pair of local state diagrams can be uniquely derived. This can be done by simply distinguishing the two cases, shown in Figure 3, of a combined state diagram transition. We propose to add a cross mark (X) to each transition which is to be interpreted as a single reception transition. Thus, assumming the local state diagram specification of X.75 in Figure 4 captures the concept intended by the X.75

designer(s) (as we believe it does), the corresponding modified combined state diagrams should be as shown in Figure 6.

In the previous discussion we have simply viewed a combined state diagram in the X.75 document as a superposition of a related pair of local state diagrams, i.e. as a concise specification from which the local state diagrams can be derived. We did not attach any physical meaning to a state (or node) in such a combined state diagram. However, according to the X.75 document [2], such a state is interpreted as a state of a logical channel. We find this interpretation confusing for the reasons which follow. Indeed, as a logical channel logically interconnects two STEs for the exchange of information pertaining to a virtual circuit, a logical channel state must have the property that both STEs at any time see the same state of the logical channel. But this 'global' property of a logical channel state simply does not exist in a combined state diagram specification, as demonstrated by the following example. Looking at the Call Establishment procedure in Figure 2b, we see that the transmission of a Call Request packet by STE-X on a "logical channel" in the Ready state (p1) causes the "logical channel" to enter the STE-X Call Request state (p2). As soon as such a transmission is initiated, process STE-X enters the 'STE-X Call Request' state. At that instant, process STE-Y views the "logical channel" to be in the 'Ready' state (p1); it will only enter the state described by p2 when it receives the Call Request packet. Process STE-Y may, however, enter the 'STE-Y Call Request' state (p3) if it transmits a Call Request packet before receiving the Call Request packet from STE-Y. So, when the "logical channel" is said to be in the 'STE-X Call Request' state (p2), process STE-X views the "logical channel" to be in the 'STE-X Call Request' state (p2), but process STE-Y may view the "logical channel" to be in one of the following possible states: i) the 'Ready' state (r1), ii) the 'STE-X Call Request' state(p2), or iii) the 'STE-Y Call Request' state (p3). Hence, if we view the state in a combined state diagram as the state of a "logical channel" then the interpretation of this state is often ambiguous (states in a combined state diagram represent just states of an individual process STE). To avoid such ambiguity, we can represent a state of the logical channel by a pair of states of the combined state diagram, e.g. (p2,r1), (p2,p2), and (p2,p3); The first identifier in the pair indicates the state of STE-X, and the second identifier, the state of STE-Y. If we want to describe actions in an STE without imposing any actions on the other STE, we can still refer to states of the logical channel by simply using a special symbol, say '*', in place of the state identifier of the other STE. For example, (p2,*) represents the state of the logical channel when STE-X is in state p2, irrespective of the state of STE-Y (whether STE-Y is in r1, p2, or p3 is irrelevant).

V. OUTLINE OF THE VALIDATION PROCEDURE

The procedure used to validate the X.75 specification, namely validation via resynthesis, is based on the synthesis rules developed by Zafiropulo *et al.* [1], which can handle the following types of errors:

i) *unspecified receptions:* An unspecified reception occurs when a positive arc that can be traversed is missing, in other words when a reception that can take place is not specified in the design. Unspecified receptions are harmful because the implication of an unspecified reception is that the respective process enters an unknown state via a transition not specified in the design. As a consequence, the occurrence of an unspecified reception means the subsequent behavior of the interaction is unpredictable

ii) *state ambiguities:* The pair (x,y) is said to be a stable state pair if a state x in one process and a state y in the other can be reached with no packet in flight. In such a case, states x and y coexist (stably) until the next transmission occurs. Now, a state ambiguity exists when a state in one process can coexist stably with more than one state in the other process. State ambiguities do not necessarily represent errors, but they are well worth examining as an indicator of potential errors.

iii) *state deadlocks:* A state deadlock occurs when each and every processes has no alternative but to remain indefinitely in the same state. So, a state deadlock is present when no transmissions are possible from the current stable state of each process.

iv) *nonexecutable interactions:* A nonexecutable interaction is present when a design creates arcs that can never be traversed (i.e. the receptions specified cannot occur) under normal operating conditions. A nonexecutable interaction is equivalent to dead code in a computer program.

This synthesis procedure is based on a set of three production rules that create only those arcs needed to prevent unspecified receptions (the reader is referred to [1] for a detailed description of the production rules and the synthesis procedure). Starting from the initial state, the protocol designer must specify the transmission transitions and the destination state for each transition, based on the semantics of the function he is trying to implement. The reception transitions are generated mechanically by applying the production rules repeatedly for each transmission arc created. Thus, at the completion of the synthesis process, the protocol specification is guaranteed to contain no unspecified receptions and nonexecutable interactions. State ambiguities and state deadlocks can also be detected by maintaining a table of stable state pairs. A test for state ambiguity can be performed every time a new pair of stable states is generated. A state ambiguity occurs if there are two or more pairs of stable states containing a common state in one process. A state deadlock is detected, however, at the end of the synthesis process by the presence of a pair of stable states with no outgoing transmission arc.

For validation, we simply apply the synthesis procedure described above to generate a good duplicate copy of the original protocol, i.e. a protocol specification which corresponds to the original one, but which is free from unspecified receptions and nonexecutable interactions. This can be done because the given protocol specification provides us with the transmission transitions and the destination states (for each transition) to be specified in the synthesis process. Thus, a protocol can be validated against unspecified receptions and nonexecutable interactions by simply comparing the original protocol specification with the synthesized one. A protocol specification has no unspecified receptions and nonexecutable interactions if it is identical to the synthesized specification. Nonexecutable interactions may be detected at the completion of the entire synthesis process, by checking if there are any arcs or nodes specified in the original specification but not in the synthesized one. An unspecified reception may, however, be detected at an early step in the interactive synthesis process by checking if a generated reception arc is not specified in the original specification. Clearly, the state deadlocks and the state ambiguities can be detected in the same way as described previously in the synthesis procedure.

VI. DECOMPOSITION OF THE X.75 SPECIFICATION

The validation procedure presented above becomes less tractable when the protocol specification to be validated is large and complex. Diagnosis of detected errors may be unwieldy, and a great deal of time is likely required to generate a large number of redundant arcs in the synthesis process. To facilitate the validation process it is desirable to decompose a large specification

into a number of smaller, more manageable components.

We have applied a decomposition method to segment the X.75 specification in Figure 5 into four components as shown in Figure 7, and validated each component separately to achieve the "equivalent" result of validating the entire X.75 specification. The decomposition method used is discussed in detail in the companion paper [6] and so is only briefly presented here. This decomposition method has the property that if every component of the decomposed specification has no state ambiguity, then the entire specification has an UR if and only if a component of the decomposed specification has that UR. Moreover, if a state ambiguity exists in a component of the decomposed specification, then any UR in the decomposed specification is also an UR in the entire specification and state ambiguities in the decomposed specification are the ones from which other state ambiguities and subsequent URs in the entire specification, if any, originate, i.e. the latter state ambiguites result from the propagation of the former.

Basically, the decomposition method is based on two common structures, i) the so called nested structure and ii) the so called sequential structure, of the class of call-based protocols.

i) *The nested structure:* A protocol specification exhibits the nested structure if a number of states in the specification can be combined (into a special state called a *superstate)* so as to produce two much simpler specifications: the external component and the internal component. As illustrated in Figure 8a, the *external component* specifies the interactions among the superstate and the remaining states while the *internal component* specifies the interactions among the states within the superstate, as illustrated in Figure 8a. The rules and conditions for the nested structure decomposition and their validity are presented in the companion paper [6].

ii) *The sequential structure:* A protocol specification with some initial state S exhibits the Sequential structure if it consists of two sequential components which are connected by only a single unambiguous stable state (i.e. the interacting processes can coexist stably and unambiguously, each at this linking state). This state is the final state of the first component (having initial state S) and also the initial state of the second component. Clearly, the protocol specification can be decomposed into these two sequential components, as illustrated in Figure 8b.

The overall structure of X.75 is illustrated in Figure 9 in which the circle represents the superstate in a nested structure and the box represents the first component of a sequential structure or the final (irreducible) component of the protocol specification. From this illustration, it is clear that the nested structure can first be applied to the X.75 specification to produce the external component "Call Clearing" as shown in Figure 7a where the superstate contains the states from 1 to 10. The internal component in this decomposition, i.e. the interaction among the states from 1 to 10, in turn, exhibits the sequential structure (with state 5 being the linking state) and can, therefore, be decomposed into two sequential components. The first component is the Call Establishment component (Figure 7b). The second component, the interaction among states 5 to 10, can be further decomposed into the Call Reset component and the Interrupt Transfer component owing to the nested structure as shown in Figures 7c & 7d respectively.

We note that this decomposed specification is very similar to the X.75 specification in Figure 4 derived directly from the combined state diagrams of the X.75 document. This similarity apparently reflects the 'modular' design of X.75. However, it should be mentioned that the few additional arcs in the decomposed specification (in Figure 7) are crucial for the validity of the X.75 validation.

VII. VALIDATION RESULTS AND DISCUSSIONS

Applying the validation procedure to the components of the decomposed specification of X.75 separately, we have found a number of protocol design errors.

- The first error detected is the unspecified receptions of the Call Request (CAR) packet in state 11 in STE-X and in state 12 in STE-Y. This happens when both STEs are in the initial state 1 and the Call Request (CAR) packet is transmitted in collision with the Clear Request (CLR) packet (using production rule 1).

- Other unspecified receptions, which occur in the Call Clearing and Call Reset components, result from the self-loop retransmissions of the Clear Request and Reset Request packets when the effect of timeout is not taken into consideration. Indeed, as explained below, the timeout self-loop retransmissions of the Clear Request packet (or the Reset Request packet) may perturb the system to an abnormal state, (1,11), (12,1), or (12,11), and subsequently give rise to the following unspecified receptions (URs):

State of STE-X	UR packet	State of STE-Y	UR packet
1	CLC	1	CLC
12	CLC CAR	11	CLC CAR

Starting in the initial state (1,1), if the STE-X transmits and retransmits the Clear Request packet while the STE-Y also transmits a Clear Request, then the STE-X will be in state 1 and the STE-Y in state 11, i.e. a state ambiguity occurs with the STEs now being in stable state (1,11). At that instant, if the STE-Y transmits a Clear Confirmation, and STE-X transmits a Call Request, then this will cause an UR of the CLC in state 1 in the STE-X and an UR of the CAR in state 11 in STE-Y. Symmetrically, we can see the occurrence of the stable state (12,1) and the subsequent UR of the CLC in state 1 in STE-Y and the CAR in state 12 in STE-X. On the other hand, if both STEs transmit and retransmit the Clear Request packet from the initial state (1,1), then stable state will be reached, giving rise to the URs of the CLC in state 12 in STE-X and state 11 in STE-Y.

It may be argued that these state ambiguities and URs could have been prevented should the timeout effect be taken into account, e.g. a timeout interval greater than the maximum packet delay can ensure that a CLR would not be retransmitted unless a CLR or CLC gets lost. However, in the case where a CLC packet gets lost, perturbation from the normal interaction may force the STEs to loop back and forth between the two states (11,1) and (1,12) with no progress. Indeed, in state (11,1), STE-Y may transmit a CLR which arrive at the STE-X before STE-X timeouts to retransmit a CLR (the timeout interval is long). The STEs are subsequently in state (1,12). A CLR may then be transmitted by STE-X to get the STEs back to state (11,1). The timeout interval for STE-Y is reset again; and so the system of two STEs may flip flop forever between states (11,1) and (1,12), causing tempo blocking.

Similar results are obtained for the Call Reset component. We found the following URs:

State of STE-X	UR packet	State of STE-Y	UR packet
5	REC	5	REC
10	REC INT DAT FLC	9	REC INT DAT FLC

It is important to point out, however, that retransmissions of the CLR and RER packets (in state p6 (or p7) and state d2 (or d3) as specified in the combined state diagrams of Figure2c and 2d respectively) are only optional according to the action table description in Annex 4 of the X.75 document [2]. One can alternatively choose to wait indefinitely in state p6 (or p7) for a CLR or CLC packet, and to signal a CLR packet in state d2 (or d3). Obviously, the UR problem discussed above vanishes, should this alternative be adopted.

- The last set of errors are found in the Interrupt component when we allow for further Interrupt transmission and reception. Without this provision, the specification of the Interrupt component as derived strictly from the combined state diagram (Figure 2e) passes our validation successfully. However, as stated in subsection 3.3.5 of the X.75 document, "An STE receiving a further Interrupt packet in the time between receiving one Interrupt packet and transferring the Interrupt confirmation, may either discard this Interrupt packet or reset the virtual circuit." This clearly implies the provision of a reception and consequently the provision of a transmission of a further packet in the specification. That interaction obviously can be specified in the local state diagram specification as a self-loop transition which transmits an INT packet in state 6 in STE-X and the corresponding self-loop reception of the INT packet in state 7 in STE-Y; or similarly, the self-loop transmission and reception of an INT packet in state 7 in STE-Y and in state 6 in STE-X respectively. Perhaps a well-behaved STE is never expected to issue an Interrupt before having received the confirmation to the previous one. But the point is that if there exists such an ill-behaved STE which may issue a further Interrupt, the preventive solution prescribed in the X.75 document, discarding this Interrupt packet or resetting the virtual circuit, is simply not adequate. There is still the problem of unspecified receptions resulting from the further Interrupt as explained below.

As in the case of Call Clearing and Call Reset components, the self-loop transmission transition in state 6 of the Interrupt component gives rise to state ambiguities, e.g. (5,5) and (5,7), and URs of packet INC in state 5 and state 7. From initial state (5,5), STE-X can transmit an INT to get to state (6,6). If STE-X transmits an INT in collision with an INC transmitted by STE-Y, a perturbed state will be reached, namely (5,6). Now, STE-Y can transmit an INC which causes an UR of the INC packet in state 5 in STE-X, or STE-Y can transmit an INT and STE-X receives the INT to get to state (7,8) in which STE-Y can transmit an INC to cause an UR in state 7 in STE-X. Likewise, we have URs of the INC in state 5 and 7 in the STE-Y. We note that the state ambiguities and URs occur irrespective of the option to handle the reception of further Interrupt packet, whether discarding it or resetting the virtual circuit. So, it would be simple to avoid the problem by stating in the X.75 document that further interrupt is not allowed.

We also tried to resynthesize the entire X.75 specification to verify the above results of validating the X.75 decomposed specification. Excluding the self-loop transmissions of the CLR and RER packets, the resynthesis of the entire X.75 specification was sucessfully completed (with no error), except for the UR of the CAR mentioned earlier. But as soon as the self-loop transmission arcs are introduced in the synthesis, state ambiguities and several URs prevail. The 31 pairs

of URs found, including 8 pairs identified in the validation of the X.75 decomposed validation above, are shown in Figure 10. In figure 11 we present duologues leading to URs to explain how these URs can occur from the initial state (1,1). Many of these URs are non-trivial and they would hardly be detected without the aid of an automated tool.

VIII. CONCLUSIONS

Using a subset of the X.75 packet level protocol as an example we have demonstrated that the validation procedure together with the decomposition method we have described can be applied to a reasonably complex protocol. The results from validating this well defined protocol may not be as significant as the ones which may be obtained from validating a protocol at an earlier stage of its development. But they are interesting enough to show the usefulness of the validation procedure and the decomposition method in reducing the design time and increasing the correctness of communication protocols.

The validation experiment did not identify any errors in the X.75 specification which could result in incorrect operation of the protocol for the most probable interaction cases.

It did, however, identify an unspecified reception of the Call Request packet at each 'Clear Request' state (p2 and p3) in the Call Clearing procedure. A similar error with regard to the X.25 packet level specification has been uncovered in an early version of the X.25 specification in a submission to CCITT by 'IBM Europe' [7] and by Belnes and Lynning [BELS 77] independently and was subsequently corrected in the X.25 final version.

It further identifies several errors owing to the self-loop transitions (which "may take place after a certain timeout") in the Call Clearing and Reset procedures and points out the problem when further Interrupt (INT) packet transmission and reception are allowed in the Interrupt Transfer procedure.

We have also discussed some difficulties which arise from the combined state diagram specification of X.75 and proposed a modified combined state diagram model for the formal specification of the class of "send/receive symmetric", full-duplex protocols including the X.75 packet level protocol. This modified specification is only slightly different from the former combined state diagram specification, and yet it allows every designer implementing the protocol to simply and uniquely derive the related pair of local state diagrams for the two interacting processes.

ACKNOWLEDGEMENT

The authors would like to thank Dr. N. Naffah for suggesting the validation of X.75 and Dr. P. Zafiropulo and Dr. H. Rudin for their many helpful comments on an earlier version of this paper.

REFERENCES

[1] Zafiropulo, P., West, C., Rudin, H., Cowan, D., and Brand, D., "Toward analyzing and synthesizing protocols," IEEE Trans. Commun., vol. COM-28, pp. 651-661, April 1980.

[2] The final draft (revision 1) of the International Consulative Committee for Telegraph and Telephony (CCITT) Recommendation X.75, Geneva, Feb. 1980.

[3] West, C.H. and Zafiropulo, P., "Automated validation of a communications protocol: The CCITT X.21 Recommendation," IBM I. Res. Develp., vol. 22, pp. 60-71, Jan. 1978.

[4] Belnes, D. and Lynning, L., "Some problems with the X.25 packet level protocol," ACM Computer Communications Review, Oct. 1977.

[5] Bochmann, G.V., "Notes on the X.25 for virtual call establishment and clearing," ACM Computer Communications Review, Oct. 1977, pp. 53-59.

[6] Vuong, S.T. and Cowan, D.D., "A decomposition method for the validation of structured protocols," to be published.

[7] IBM Europe, "Technical improvements to CCITT Recommendation X.25," submission to Study Group VII, Oct. 1978.

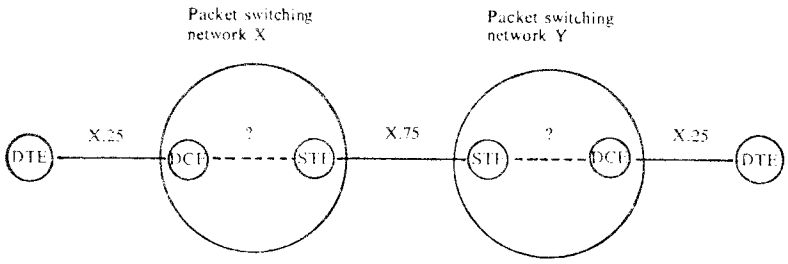
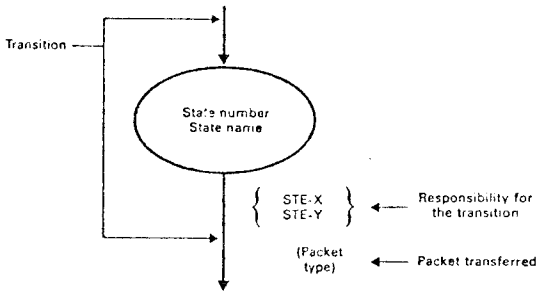


Figure 1. Example of the use of X.75 in network interconnection



Note 1. – Each state is represented by an ellipse wherein the state name and number are indicated.
Note 2. – Each state transition is represented by an arrow. The responsibility for the transition (STE-X or STE-Y) and the packet that has been transferred are indicated beside that arrow.

a. Symbol definition of the state diagrams

Figure 2. Combined state diagram specification of X.75

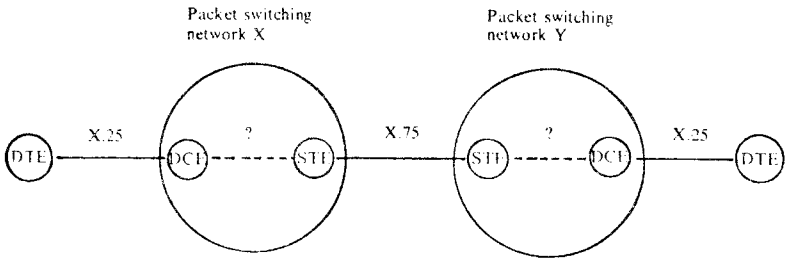
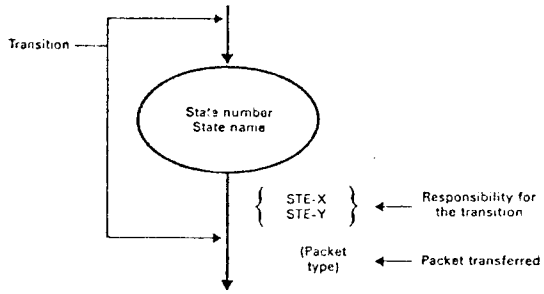


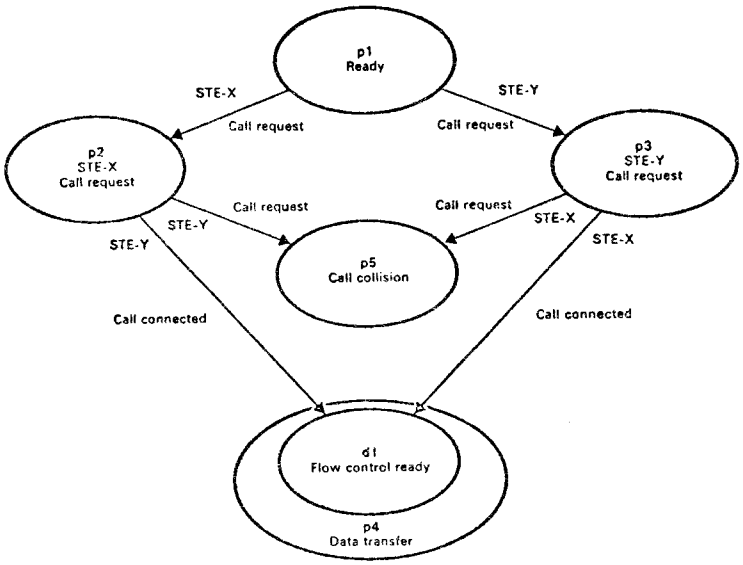
Figure 1. Example of the use of X.75 in network interconnection



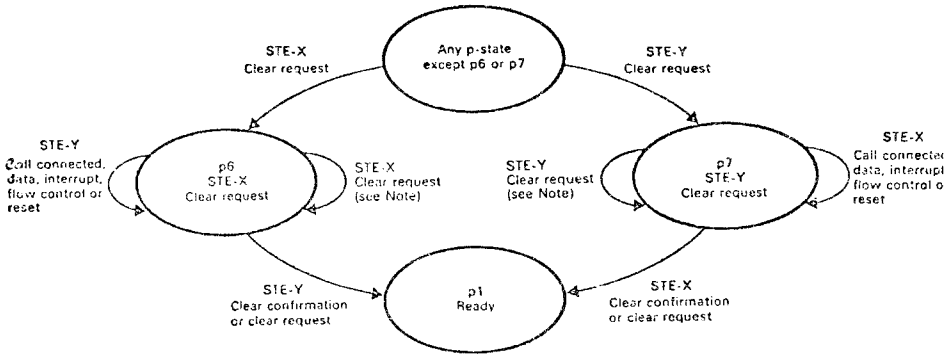
Note 1. – Each state is represented by an ellipse wherein the state name and number are indicated.
Note 2. – Each state transition is represented by an arrow. The responsibility for the transition (STE-X or STE-Y) and the packet that has been transferred are indicated beside that arrow.

a. Symbol definition of the state diagrams

Figure 2. Combined state diagram specification of X.75



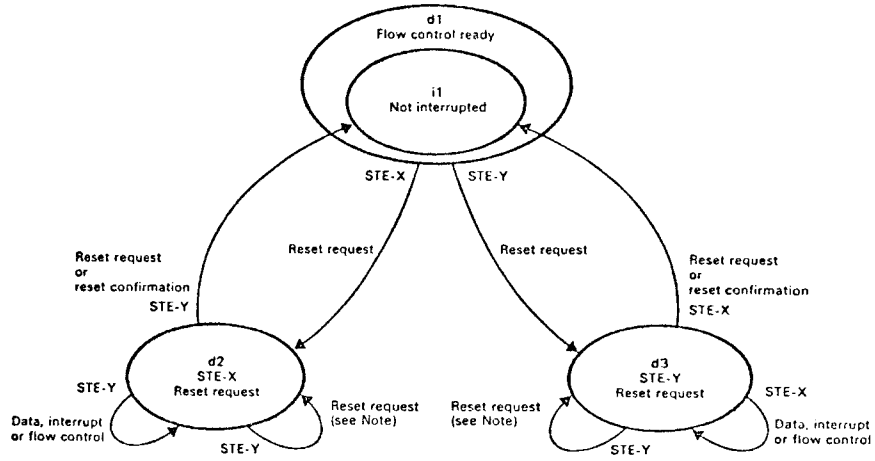
b. Call Establishment procedure



c. Call Clearing procedure

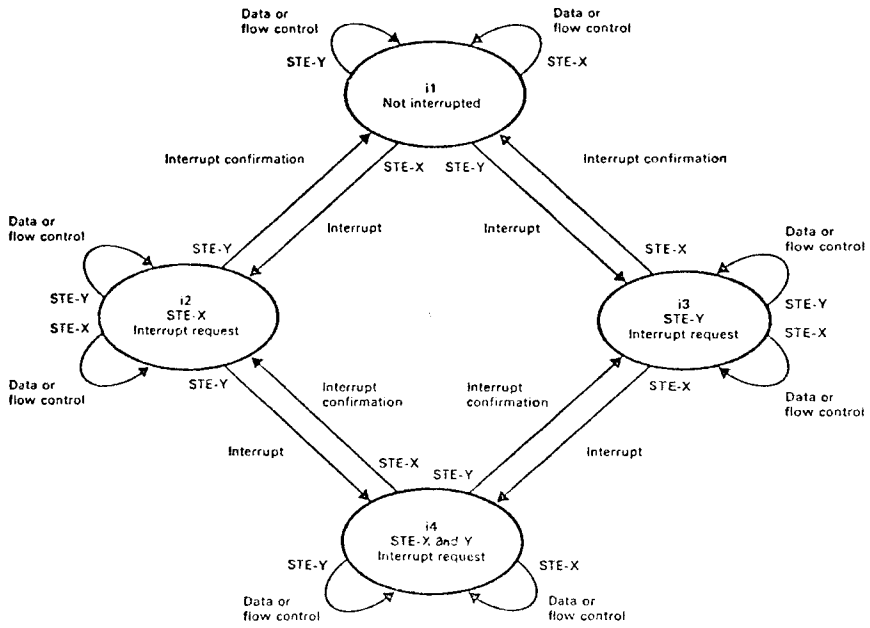
Note. – This transition may take place after a time-out.

Figure 2 (Cont.). Combined state diagram specification of X.75



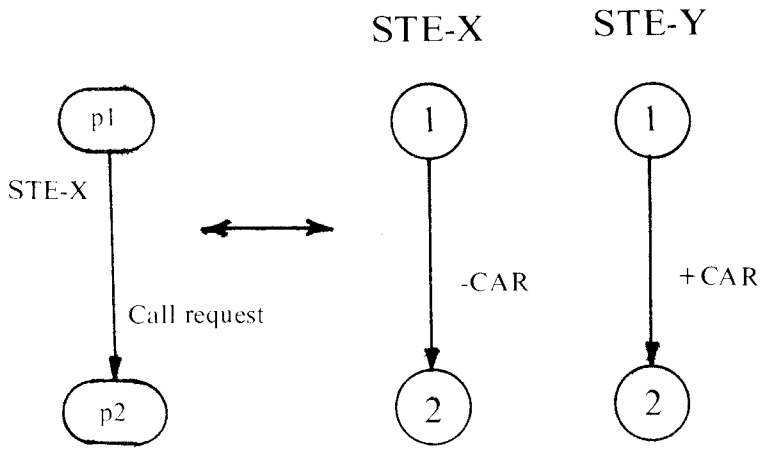
Note. – This transition may take place after a time-out.

d. Call Reset procedure

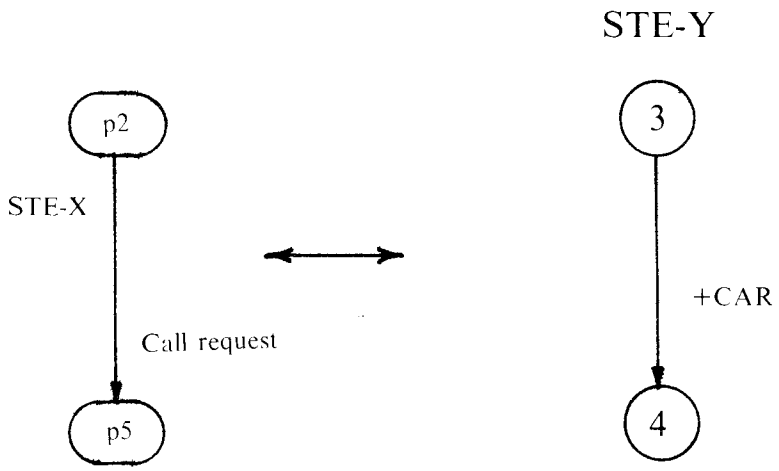


e. Interrupt Transfer procedure

Figure 2 (Cont.). Combined state diagram specification of X.75



a. Combined transition interpreted as a pair of send/receive transitions



b. Combined transition interpreted as a single reception transition

Figure 3. Example of the mapping between combined state diagram transition and local state diagram transition(s)

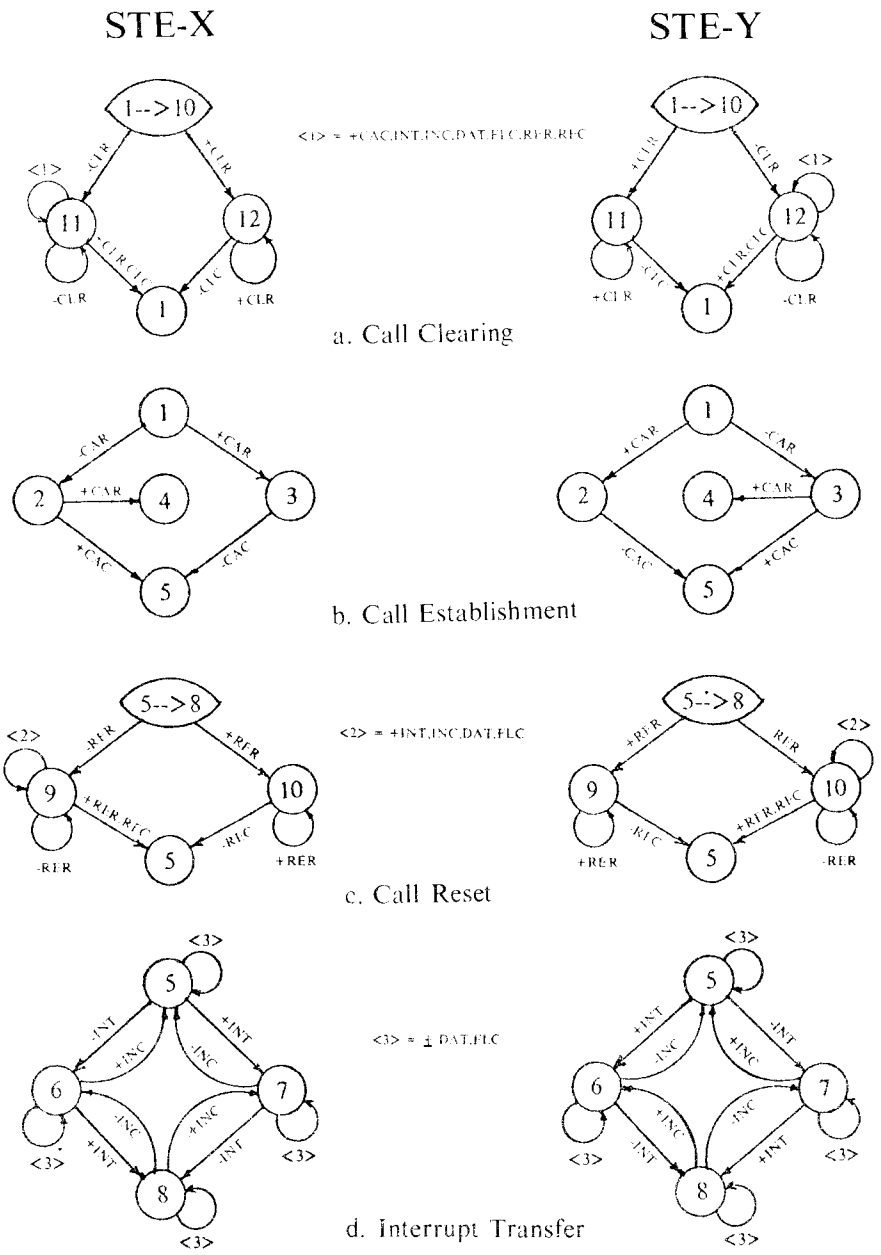
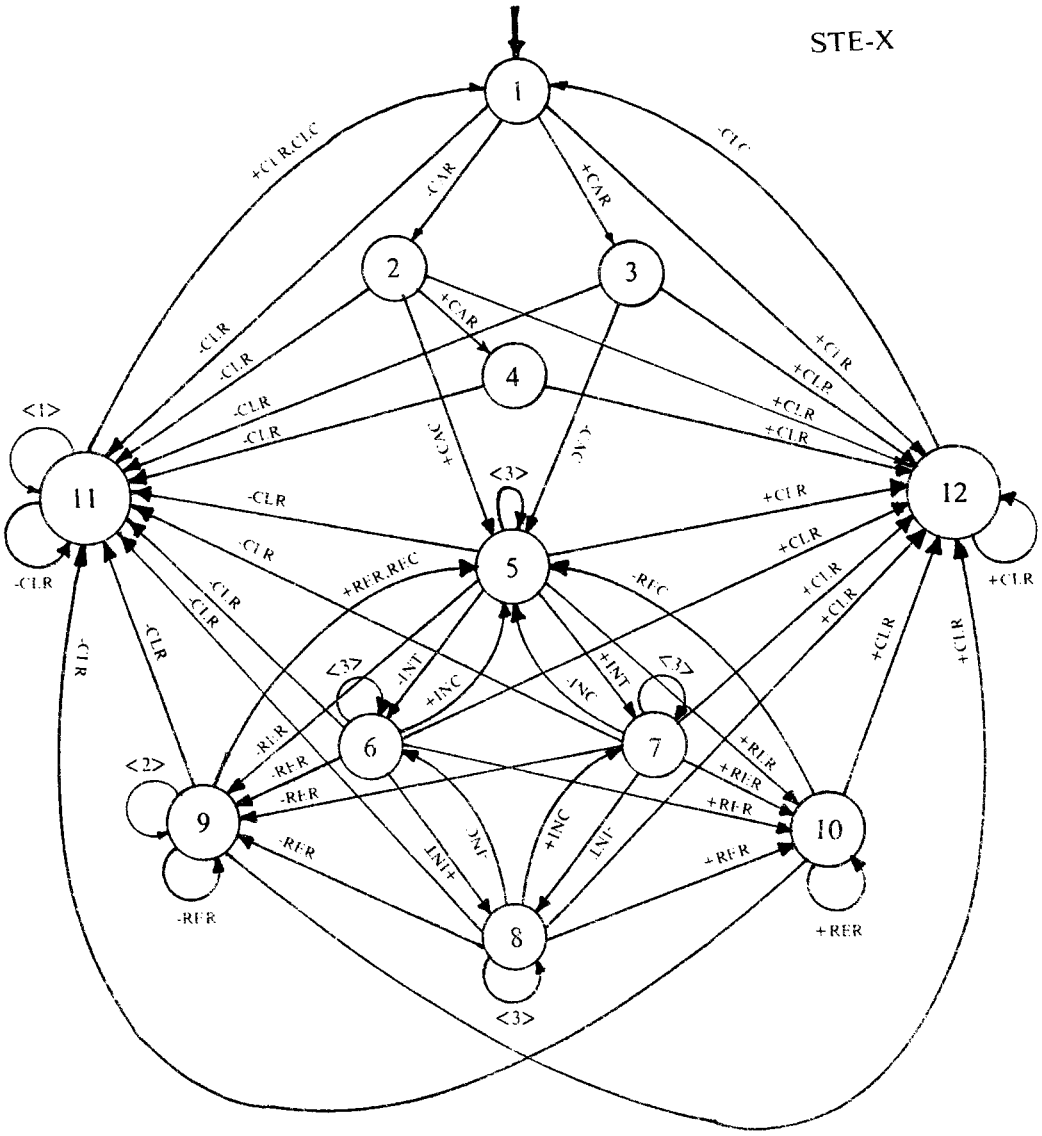


Figure 4. X.75 specification in terms of local state diagrams

EVENT (PACKET)	
Local state diagram	Combined state diagram
CAR	Call request
CAC	Call confirmation
CLR	Clear request
CLC	Clear confirmation
INT	Interrupt
INC	Interrupt confirmation
DAT	Data
FLC	Flow control
RER	Reset request
REC	Reset confirmation

STATE	
Local state diagram	Combined state diagram
1	Ready (p1)
2	STE-X Call Request (p2)
3	STE-Y Call Request (p3)
4	Call Collision (p5)
5	Not Interrupted (i1)
6	STE-X Interrupt Request (i2)
7	STE-Y Interrupt Request (i3)
8	STE-X & STE-Y Interrupt Request (i4)
9	STE-X Reset Request (d2)
10	STE-Y Reset Request (d3)
11	STE-X Clear Request (p6)
12	STE-Y Clear Request (p7)

Legends for Figures 4 & 7

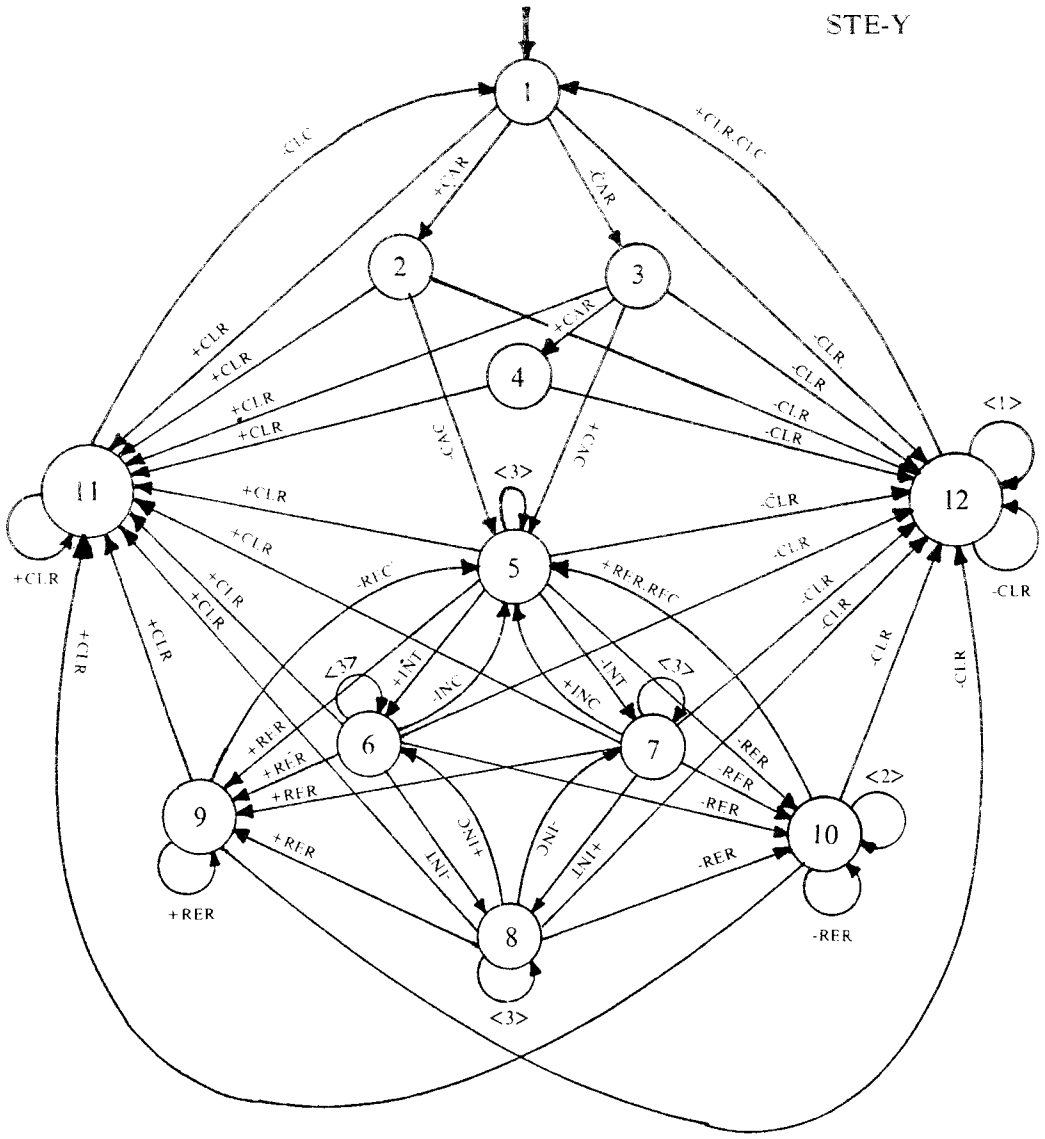


<1> = +CAC,INT,INC,DAT,FLC,RER,REC

<2> = +INT,INC,DAT,FLC

<3> = ± DAT,FLC

Figure 5a. Specification of X.75 (STE-X)

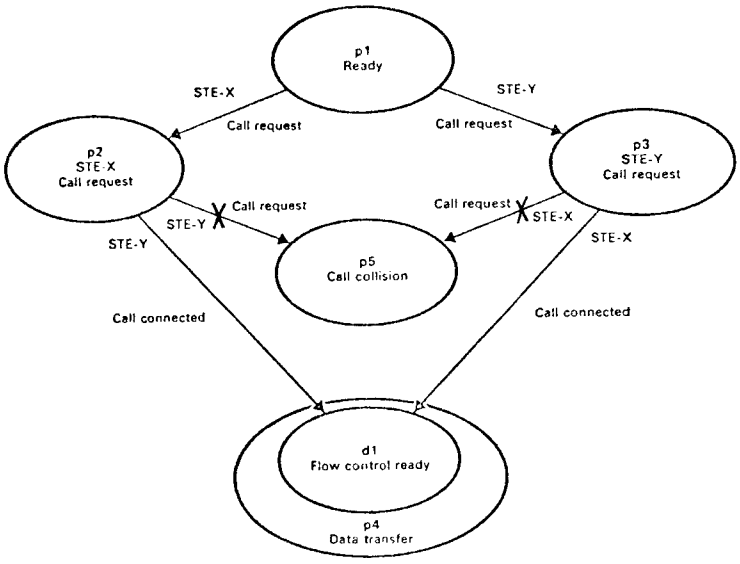


<1> = +CAC.INT.INC.DAT.FLC.RER.REC

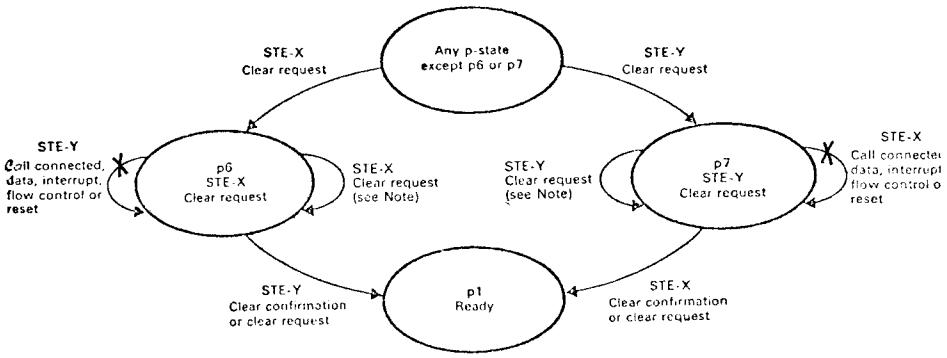
<2> = +INT.INC.DAT.FLC

<3> = ± DAT.FLC

Figure 5b. Specification of X.75 (STE-Y)



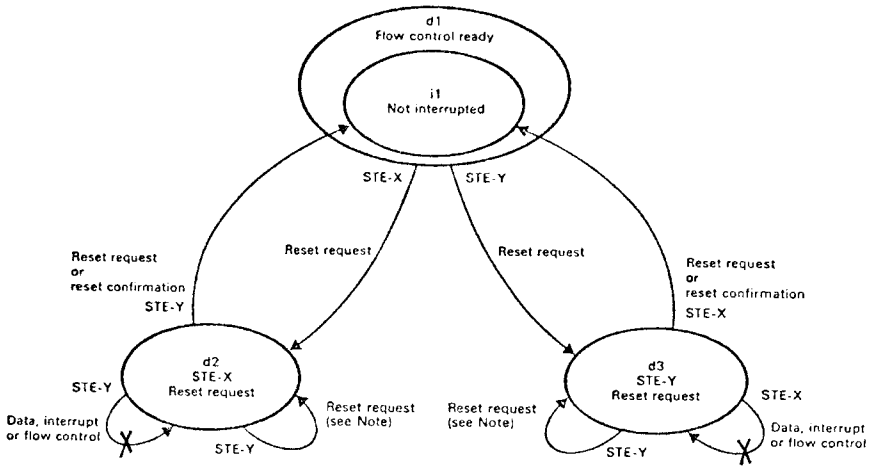
b. Call Establishment procedure



c. Call Clearing procedure

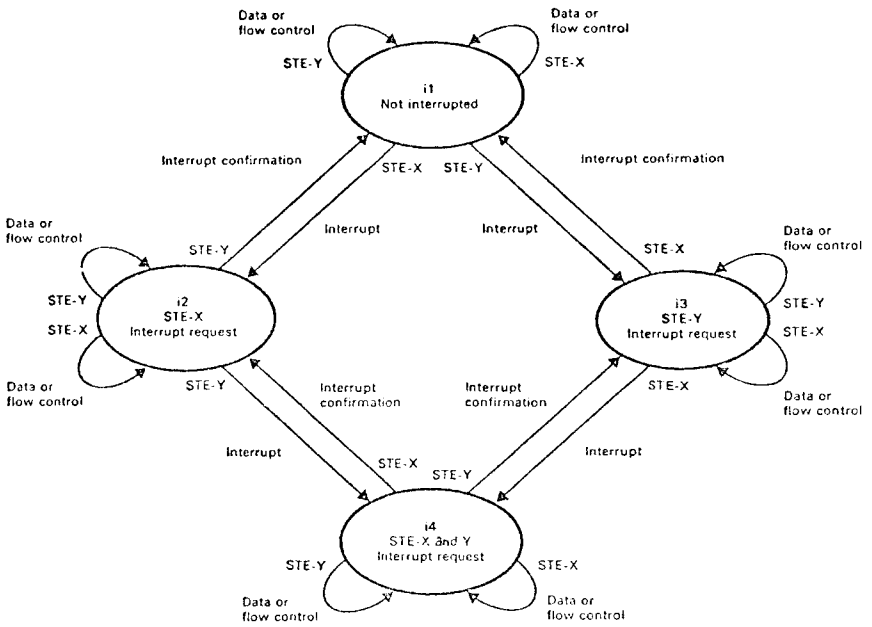
Note. – This transition may take place after a time-out.

Figure 6. Modified specification of X.75



Note. - This transition may take place after a time-out.

d. Call Reset procedure



e. Interrupt Transfer procedure

Figure 6 (Cont.). Modified specification of X.75

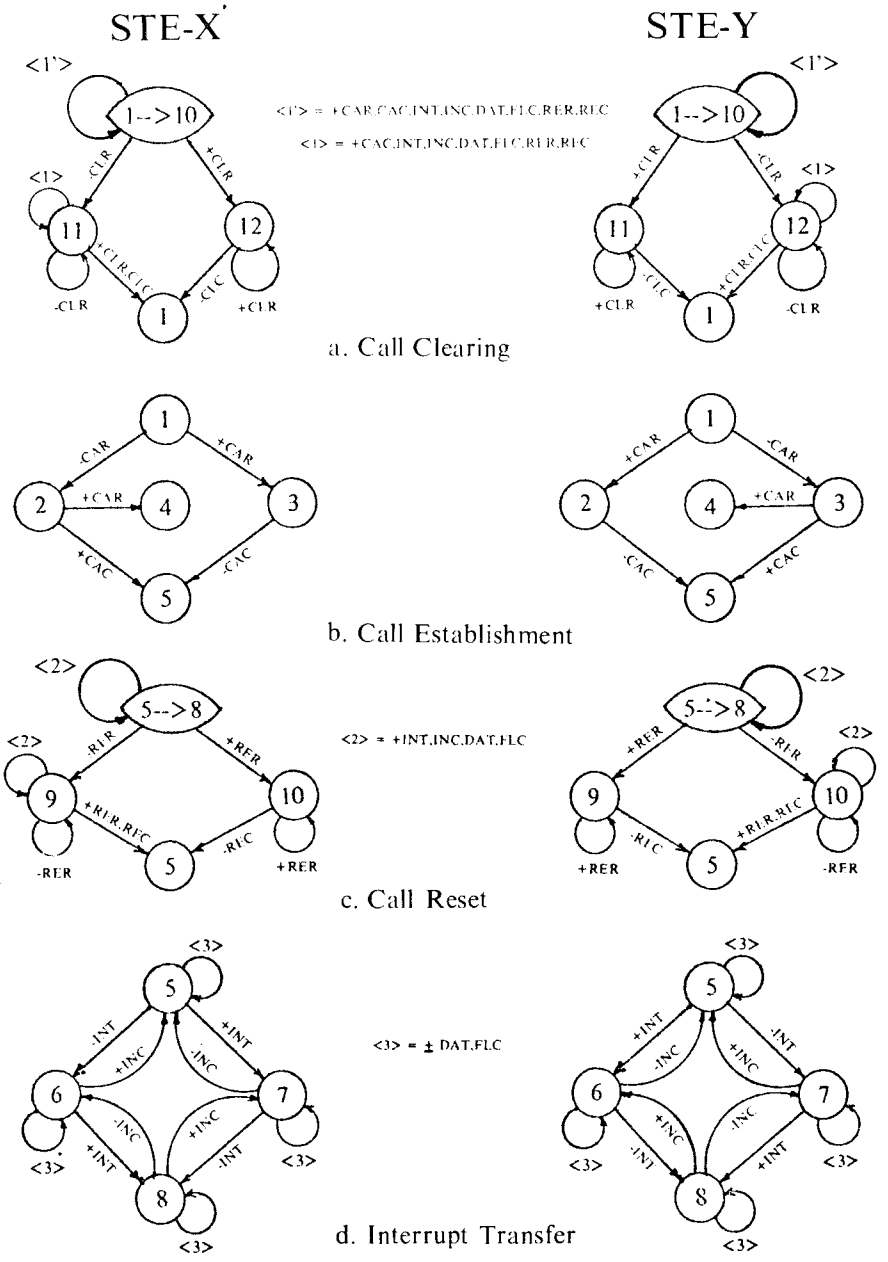
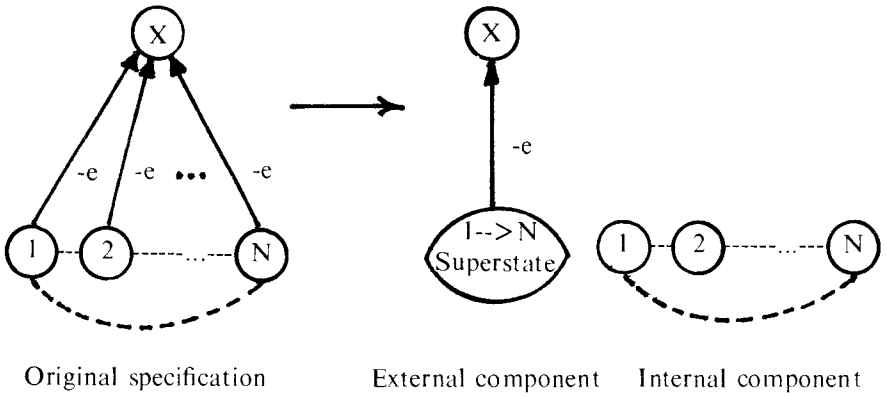
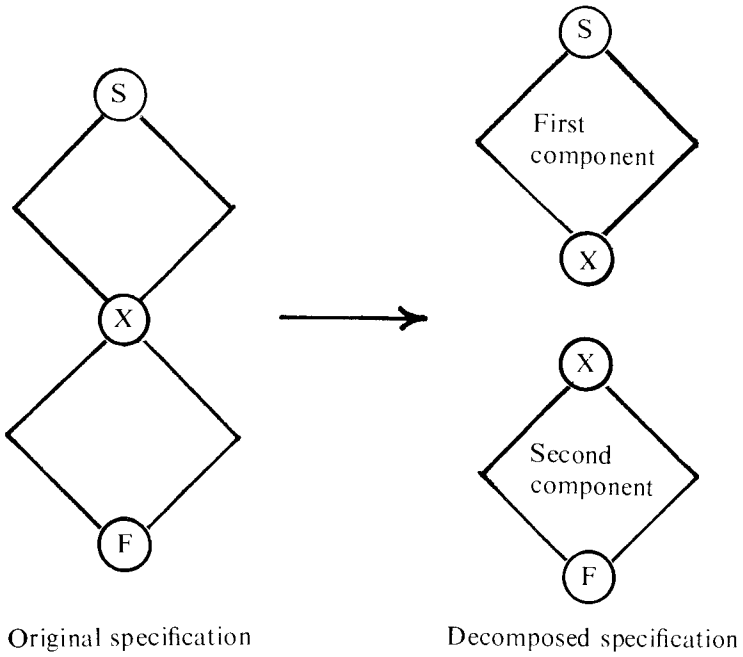


Figure 7. Decomposed specification of X.75



a. Decomposition on the nested structure



b. Decomposition on the sequential structure

Figure 8. Illustration of the decomposition method

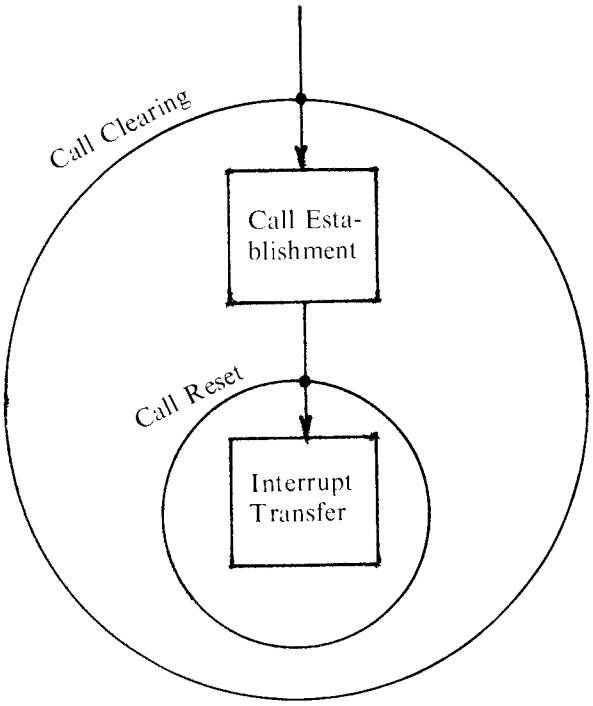


Figure 9. Structure of X.75

State of STE-X	UR packet	Section in Figure 11 explaining this UR
1	CLC	a
	CAC	b
2	CLC	b
3	CLC	c
	RER	e
	INT	e
	DAT	e
	FLC	e
4	CLC	c
	RER	e
	INT	e
	DAT	e
	FLC	e
5	CLC	c
	REC	f
	INC	g
6	CLC	c
	REC	f
7	REC	f
	INC	g
8	REC	f
	INC	g
9	CLC	c
10	REC	f
	INT	g
	INC	g
	DAT	g
	FLC	g
12	CLC	a
	CAR	d
	CAC	d

Figure 10. Unspecified receptions (URs) in the X.75 specification - for STE-X (symmetrically for STE-Y)

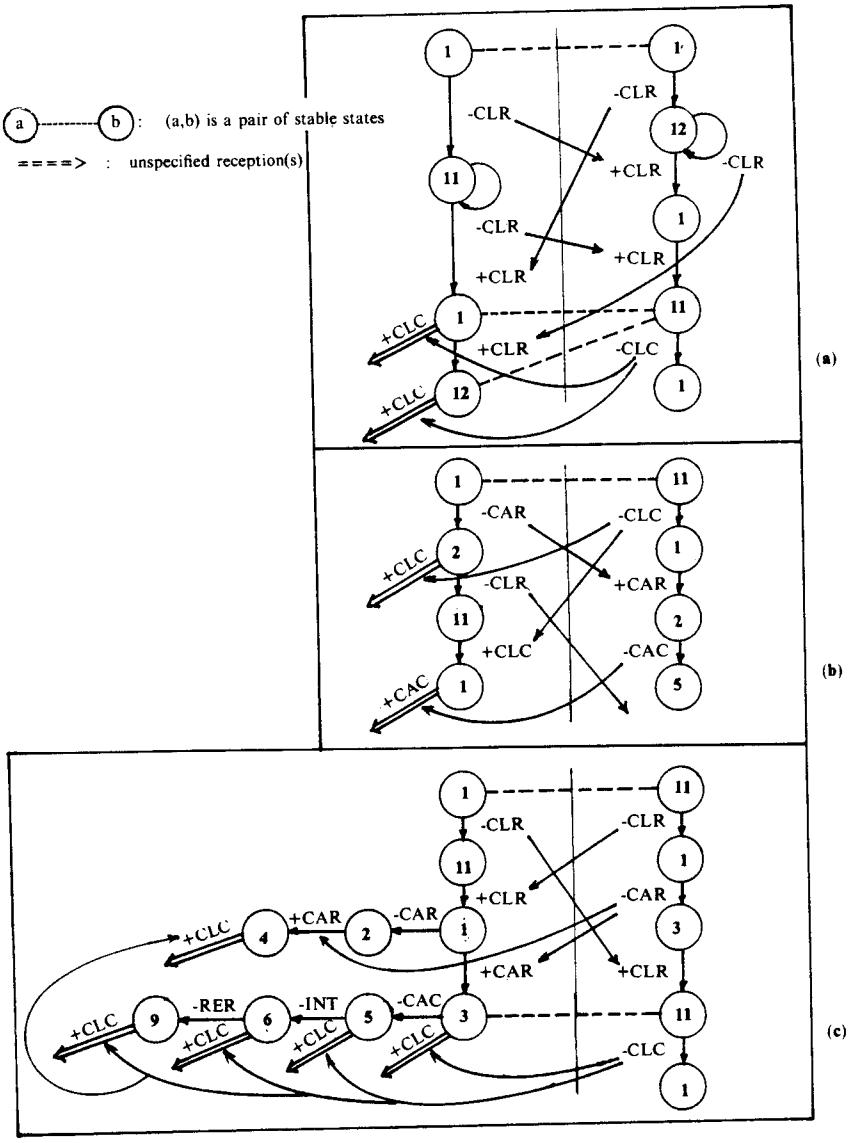
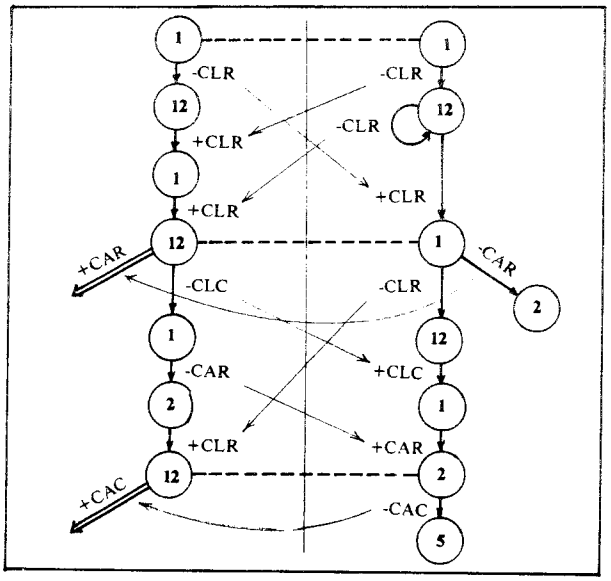
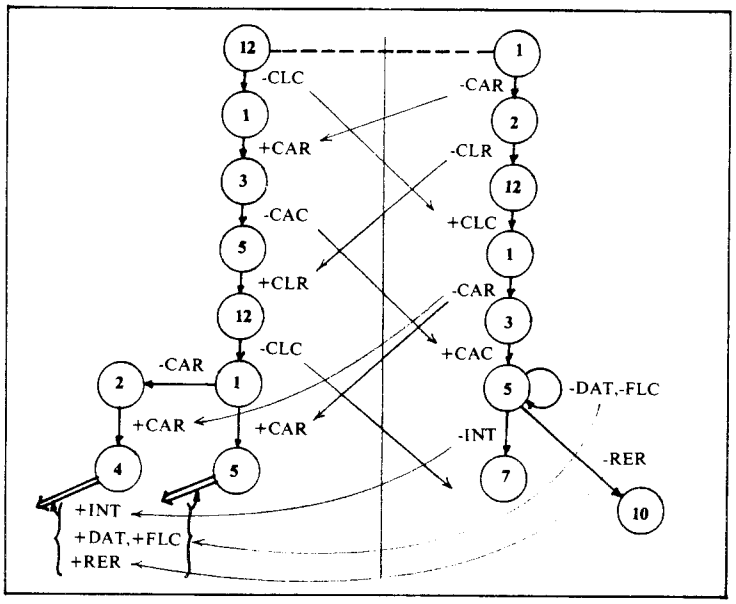


Figure 11. Interactions leading to unspecified receptions in the X.75 specification (owing to CLR retransmissions)



(d)



(e)

Figure 11 (cont.). Interactions leading to unspecified receptions (owing to CLR retranmissions)

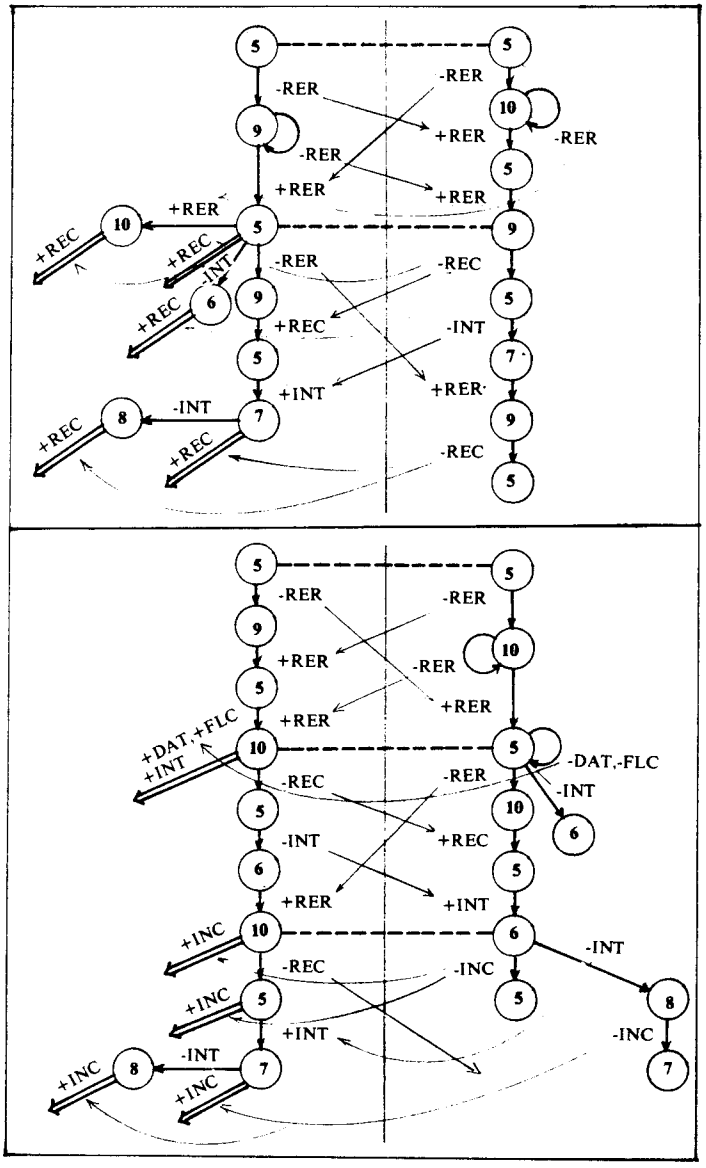


Figure 11 (cont.). Interactions leading to unspecified receptions (owing to RER retransmissions)