 Compact Modeling and Rapid Prototyping of Communication Software with ECATNets: A Case Study

M. Bettaz, M. Maouche and M. Soualmi
Institute of Computing Science
University of Constantine
Constantine, 25000 Algeria

M. Boukebeche
Institute of Computing Science
University of Constantine Algeria
& Laboratoire MASI Paris 6 France

Abstract
This paper deals with using formal description techniques in the early stage of the life-cycle of computer communication software in concordance with the ISO/OSI standards. The suggested approach allows to consider the software under design as a set of building blocks which may be easily put together, while the proposed formalism allows for compact modeling and rapid prototyping of such blocks. Our methodology seems to be general enough, since it has been applied to different case studies. The style of the paper is kept intentionally informal in order to emphasize the practical contribution of the work rather than its theoretical one.

Keywords and Phrases
Communication standards, modeling and specification, rapid prototyping, safety verification

Introduction

Let us begin this section by stating that by communication software we mean the layer facilities (i.e. protocols and services) which may be fully or partly implemented by software components, regardless of any hardware consideration.

ECATNets are a form of algebraic high-level net model which combines the strength of Petri nets with that of abstract data types. Their formal definition, motivation and relation to other works are given in (Bettaz et al. 1992a). Two algebraic structures characterize ECATNets. The first one, related to the syntactic notations, labelling the net, aims to design models with more compactness, since it is defined on (equivalence classes of) algebraic terms. The second structure, used to define semantics of the net, allows for rapid prototyping of the achieved models, since it is given in terms of rewrite logic (Meseguer 1991).

A major objective of this paper is to show the generality of our formalism by applying it to different case studies in conformity with the OSI environment.

The paper is organized in the following way. In section 2 we review some basic notions about ECATNets. In section 3 we present our case study. The studied protocol is a symmetric data-link layer protocol using the selective retransmission strategy. For lack of space some details of this study are omitted. A complete case study may be found in (Bettaz et al. 1992b). In section 4 we show, through a scenario, how to run the protocol prototype. In section 5 we end by giving some concluding remarks.

ECATNets

The graphical representation of a generic ECATNet is given by Fig.1. IC, DT and CT are multisets of (equivalence classes of) terms, with \( \Theta, \cap, C, \setminus \) being respectively the multiset union, intersection, inclusion and difference, and \( \phi_M \) the identity element. We let \([x]_\Theta\) denote the equivalence class of \( x \) w.r.t. the ACI (Associative, Commutative and with Identity element) axioms for \( \Theta \). The terms are defined on a user declared algebraic specification of an abstract data type. We let \([x]_\Theta\) or just \([x]\) denote the equivalence class of \( x \) w.r.t. the axioms (equations) given by the user in his/her specification.

TC is a boolean term which may contain variables occurring in IC, DT and CT. Each place is associated a capacity C(p) defined as a multiset of closed (equivalence classes of) terms. The marking M(p) of a place p of the net, which is itself a multiset of closed terms, is defined w.r.t. the capacity (which may be infinite).

Fig.1 A generic ECATNet

A transition t is fireable when various conditions are simultaneously true. The first condition is that every IC(p,t) for each input place p is enabled. The second condition is that TC(t) is true. Finally the addition of CT(p,t) to each output place p must not result in p exceeding its capacity when this capacity is finite. When t is fired DT(p,t) is removed from the input place p and simultaneously CT(p,t) is added to the output place p.
Transition firing and its conditions are expressed by rewrite rules which are strongly depending on the form of the syntactic notation used for representing IC. Those rewrite rules together with a set of deduction rules define a rewrite logic which gives the semantics of the net. The left-hand and right-hand sides of the rewrite rules are multiset of pairs of the form (p, m[]p), where p is a place of the net, \( \otimes \) the multiset union, and \( \phi_p \) the identity element for this case. Let us recall in the following part of this section the forms of the rewrite rules to associate with the transitions of a concrete ECATNet. The used procedure is motivated in (Bettaz et al. 1992a).

1) IC(p,t) is of the form \([m]p\)

   case 1: \([IC(p,t)]p = [DT(p,t)]p\)

   \( t: (p, [IC(p,t)]p) \rightarrow (p', \text{CT}(p', t')]p) \)

   case 2: \([IC(p,t)]p \cap [DT(p,t)]p = \phi_p \)

   \( t: (p, [IC(p,t)]p) \otimes (p, [DT(p,t)]p) \cap [M(p)]p \rightarrow (p, [IC(p,t)]p) \otimes (p', \text{CT}(p', t')]p) \)

   Case 3: \([IC(p,t)]p \cap [DT(p,t)]p \neq \phi_p \)

   This case may be solved in an elegant way by remarking that it could be brought to the two already treated cases (Bettaz and Maouche 1991).

2) IC(p,t) is of the form \([-m]p\)

   \( t: (p, [DT(p,t)]p) \cap [M(p)]p \rightarrow (p', \text{CT}(p', t')]p) \) if \([\{IC(p,t)]p \cap [M(p)]p = \phi_p \} \rightarrow \text{false} \)

3) IC(p,t) = empty

   \( t: (p, [DT(p,t)]p) \cap [M(p)]p \rightarrow (p', \text{CT}(p', t')]p) \) if \([M(p)]p \rightarrow \phi_p \)

   When the place capacity \(C(p)\) is finite, the conditional part of our rewrite rule will include the following component:

   \( ([\text{CT}(p,t)]p \otimes [M(p)]p) \cap [C(p)]p \rightarrow [\text{CT}(p,t)]p \otimes [M(p)]p) \) (Cap)

   In the case where there is a transition condition \(TC(t)\) the conditional part of our rewrite rule will include the following component: \([TC(t)]p \rightarrow \text{true}\).

Comment: if one or more output place(s) has a finite capacity, the conditional part of the rewrite rule has to include a component of the form denoted by (Cap) for each one of these places.

THE CASE STUDY

In the following we will present the main standard actions of our protocol. These actions are specific to the data transfer phase. For the primary station we will describe the following actions:

Iframe sending. Expected ACK-frame received, I-frame retransmission, send-window edge updating. For the secondary station we will describe the following actions: Expected I-frame received, Receive-window edge updating. For the channel we will describe the following action: Lose I-frame.

The algebraic specifications involved in the ECATNet syntactic notations are in reality simple, and most of them are often available as basic library standards. The reader interested in such specifications may consult (Bettaz et al. 1992a), where some examples are given in an OBJ-like notation.

Primary Station (P)

I-frame Sending

![Diagram of I-frame sending]

Informal comment

The user requests an LSDU transmission by passing it to the sender through the access point represented by FROM_USER. P constructs an I-frame from the user supplied LSDU by appending to it the frame type DT, and a sender sequence number (available in VS), then the V(S) is incremented and the I-frame is sent, provided that the send-window and the transmission medium capacity will not be exceeded. At the same time a timer is started (in TIMER), and a copy of this frame is retained (in R_LIST).

Rewrite rules

DT-FRAMING: (FROM_USER, [DATA]) \( \otimes (VS, [NS]) \rightarrow (VS, [addw(NS, 1)]) \) \( \otimes (DT\_SEND, [\langle DT, NS, DATA \rangle]) \)

DT-SENDING: (DT\_SEND, [\langle DT, NS, DATA \rangle]) \( \otimes (\langle DT, TO\_SEND, [\langle DT, NS, DATA \rangle] \rightarrow DT\_SENDING)) \)

\( \otimes (\langle DT, TO\_SEND, [\langle DT, NS, DATA \rangle]) \rightarrow DT\_SENDING)) \)

THE CASE STUDY

In the following we will present the main standard actions of our protocol. These actions are specific to the data transfer phase. For the primary station we will describe the following actions:

Iframe sending. Expected ACK-frame received, I-frame retransmission, send-window edge updating. For the secondary station we will describe the following actions: Expected I-frame received, Receive-window edge updating. For the channel we will describe the following action: Lose I-frame.

The algebraic specifications involved in the ECATNet syntactic notations are in reality simple, and most of them are often available as basic library standards. The reader interested in such specifications may consult (Bettaz et al. 1992a), where some examples are given in an OBJ-like notation.

Primary Station (P)

I-frame Sending

![Diagram of I-frame sending]

Informal comment

The user requests an LSDU transmission by passing it to the sender through the access point represented by FROM_USER. P constructs an I-frame from the user supplied LSDU by appending to it the frame type DT, and a sender sequence number (available in VS), then the V(S) is incremented and the I-frame is sent, provided that the send-window and the transmission medium capacity will not be exceeded. At the same time a timer is started (in TIMER), and a copy of this frame is retained (in R_LIST).

Rewrite rules

DT-FRAMING: (FROM_USER, [DATA]) \( \otimes (VS, [NS]) \rightarrow (VS, [addw(NS, 1)]) \) \( \otimes (DT\_SEND, [\langle DT, NS, DATA \rangle]) \)

DT-SENDING: (DT\_SEND, [\langle DT, NS, DATA \rangle]) \( \otimes (\langle DT, TO\_SEND, [\langle DT, NS, DATA \rangle] \rightarrow DT\_SENDING)) \)

\( \otimes (\langle DT, TO\_SEND, [\langle DT, NS, DATA \rangle]) \rightarrow DT\_SENDING)) \)

THE CASE STUDY

In the following we will present the main standard actions of our protocol. These actions are specific to the data transfer phase. For the primary station we will describe the following actions:

Iframe sending. Expected ACK-frame received, I-frame retransmission, send-window edge updating. For the secondary station we will describe the following actions: Expected I-frame received, Receive-window edge updating. For the channel we will describe the following action: Lose I-frame.

The algebraic specifications involved in the ECATNet syntactic notations are in reality simple, and most of them are often available as basic library standards. The reader interested in such specifications may consult (Bettaz et al. 1992a), where some examples are given in an OBJ-like notation.

Primary Station (P)

I-frame Sending

![Diagram of I-frame sending]

Informal comment

The user requests an LSDU transmission by passing it to the sender through the access point represented by FROM_USER. P constructs an I-frame from the user supplied LSDU by appending to it the frame type DT, and a sender sequence number (available in VS), then the V(S) is incremented and the I-frame is sent, provided that the send-window and the transmission medium capacity will not be exceeded. At the same time a timer is started (in TIMER), and a copy of this frame is retained (in R_LIST).

Rewrite rules

DT-FRAMING: (FROM_USER, [DATA]) \( \otimes (VS, [NS]) \rightarrow (VS, [addw(NS, 1)]) \) \( \otimes (DT\_SEND, [\langle DT, NS, DATA \rangle]) \)

DT-SENDING: (DT\_SEND, [\langle DT, NS, DATA \rangle]) \( \otimes (\langle DT, TO\_SEND, [\langle DT, NS, DATA \rangle] \rightarrow DT\_SENDING)) \)

\( \otimes (\langle DT, TO\_SEND, [\langle DT, NS, DATA \rangle]) \rightarrow DT\_SENDING)) \)
Fig. 3 Expected ACK-frame

Informal comment

A received expected ACK-frame is passed to P (via ACK_EXPECTED) to be treated. If its N(R) is equal to LWEp then the send-window edges must be increased by one (transition AK_EXPECTED1). Otherwise we must save the N(R) of this ACK-frame (in LIST) in order to update subsequently the send-window edges (transition AK_EXPECTED2). The received ACK-frame can be in-sequence, out-of-sequence, or duplicated. However, only the in-sequence case is treated in this paper (Fig. 4).

Rewrite rules

AK_EXPECTED1: (ACK_Q, [<AK,NR >]) ⊗ (P_WINDOW_EDGE, [LWEp,UWEp]) → (ACK_EXPECTED, [<AK,NR >]) ⊗ (P_WINDOW_EDGE, [LWEp,UWEp]) if [NR = LWEp] → [true]

AK_EXPECTED2(ACK_Q, [<AK,NR >]) ⊗ (P_WINDOW_EDGE, [LWEp,UWEp]) ⊗ (P_WINDOW_EDGE, [LWEp,UWEp]) → (ACK_EXPECTED, [<AK,NR >]) ⊗ (P_WINDOW_EDGE, [LWEp,UWEp]) ⊗ (LIST,[NR]) if [(NR > LWEp) and (NR < UWEp)] → [true]

Fig. 4 In-sequence ACK-frames

Informal comment

On receipt of an ACK-frame in sequence, the associated timer is reset, and the corresponding I-frame is removed from the retransmission list (transition ACKING).

Rewrite rules

ACKING: (ACK_EXPECTED, [<AK,NR >]) ⊗ (TIMER, [<t,NR >]) ⊗ (R_LIST, [<DT,NS,DATA >]) → [NR]

I-frame retransmission

This action is started if one of the following events occurs: timeout expires, reception of an acknowledgement out-of-sequence. We will present only this last case.

Fig. 5 Reception of an acknowledgement out-of-sequence

Informal comment

The retransmission action is initiated on reception of an ACK-frame out-of-sequence (available in ACK_EXPECTED). Then the sender must retransmit one (or more) I-frame which has been transmitted before the I-frame being acknowledged, and for which the ACK-frame has not been received. At the same time the associated timer (one or more) is restarted (transition RETRANS). The retransmitted I-frame(s) is (are) removed from the retransmission list and added at the end of this list.

Rewrite rules

RETRANS: (ACK_EXPECTED, [<AK,NR >]) ⊗ (ACK_EXPECTED, [φ]) ⊗ (TIMER, [<t,NS >]) ⊗ (R_LIST, [<DT,NS,DATA >]) ⊗ (P_TO,[TO]) ⊗ (P_TO,[φ]) → (MSG_Q, [<DT,NS,DATA >]) ⊗ (TIMER, [<TO,NS >]) ⊗ (ACK_EXPECTED, [<AK,NR >]) ⊗ (R_LIST, [<DT,NS,DATA >]) ⊗ (P_TO,[TO]) if [(NS < NP) → [true]] and ([(<DT,NS,DATA >) ⊗ M(MSG_Q)] ∩ C(MSG_Q) → [true])
Send-window Edge Updating

![Diagram](image_url)

Fig.6 Send-window edge updating

Informal comment

The update of LWEp and UWEp occurs when the current value of LWEp is equal to an N(B) which was already saved (transition P_WINDOW_EDGE.UPDATE).

Rewrite rules

\[
P_{\text{WINDOW}._{\text{EDGE}}.\text{UPDATE}}: (\text{LIST}, \{\text{LWEp}\}) \otimes (P_{\text{WINDOW}._{\text{EDGE}}.\text{UPDATE}}[\text{LWEp} \to \text{UWEp}]) \rightarrow (P_{\text{WINDOW}._{\text{EDGE}}.\text{UPDATE}}[\text{LWEp} \to \text{UWEp}])
\]

3.2 Secondary Station (S)

3.2.1 Expected I-frame Received

![Diagram](image_url)

Fig.7 Expected I-frame received

Informal comment

On receipt of an expected I-frame (available in MSG.Q), S passes this frame to the user (TO_USER), puts its N(S) in the receive list (REC_LIST), increments by one the number of correctly received I-frames, finally it returns an ACK-frame for this received I-frame (transition DT_EXPECTED).

Rewrite rules

\[
\text{DT}_\text{EXPECTED}(\text{MSG.Q}[\text{DT}_\text{NS}._\text{DATA}]), (S_{\text{WINDOW}._{\text{EDGE}}.\text{UPDATE}}[\text{LWEs} \to \text{UWEs}]) \otimes (S_{\text{WINDOW}._{\text{EDGE}}.\text{UPDATE}}[\text{LWEs} \to \text{UWEs}]) \rightarrow \text{DT}_\text{EXPECTED}(\text{MSG.Q}[\text{DT}_\text{NS}._\text{DATA}]), (S_{\text{WINDOW}._{\text{EDGE}}.\text{UPDATE}}[\text{LWEs} \to \text{UWEs}])
\]

Receive-window Edge Updating

![Diagram](image_url)

Fig.8 Receive-window edge updating

Informal comment

On receipt of W correct I-frames, the receive-window edges must be updated, the receive list (REC_LIST) emptied, and the number of correctly received I-frames reset to zero (transition S_WINDOW_EDGE.UPDATE).

Rewrite rules

\[
S_{\text{WINDOW}._{\text{EDGE}}.\text{UPDATE}}: (S_{\text{REC}._\text{COUNT}}[\text{W}]) \otimes (S_{\text{WINDOW}._{\text{EDGE}}.\text{UPDATE}}[\text{LWEs} \to \text{UWEs}]) \otimes (S_{\text{WINDOW}._{\text{EDGE}}.\text{UPDATE}}[\text{LWEs} \to \text{UWEs}]) \rightarrow (S_{\text{WINDOW}._{\text{EDGE}}.\text{UPDATE}}[\text{W}]) \otimes (S_{\text{WINDOW}._{\text{EDGE}}.\text{UPDATE}}[\text{LWEs} \to \text{UWEs}]) \otimes (S_{\text{WINDOW}._{\text{EDGE}}.\text{UPDATE}}[\text{LWEs} \to \text{UWEs}])
\]

Lose Frame

![Diagram](image_url)

Fig.9 Lose I-frame

Informal comment

Any I-frame may be lost when sent on the transmission medium (transition LOSE_MSG).

Rewrite rules

\[
\text{LOSE_MSG}: (\text{MSG.Q}[\text{DT}_\text{NS}._\text{DATA}]) \rightarrow \varphi_B
\]

152
The Protocol

Once given the specifications of the different actions, the specification of the protocol itself, may be obtained by a composition on places of the specifications of these actions. It seems that the composition rule is quite simple and given by the rules composition, i.e. the set of rewrite rules to associate with the whole protocol is merely given by the union of the sets associated with the different actions. However the theoretical aspect of this problem will be addressed in a future work.

SCENARIO

In this section we present a scenario, allowing us to understand the operation of the protocol through its interpretation in terms of rewrite logic. Moreover the scenario allows us to check the correctness of the composition of the rules related to the different actions.

Let a and b be two ISDUs to transmit. P will then send the corresponding two I-frames [<DT,0,a>] and [<DT,1,b>]. Let us suppose that the first I-frame is lost. Thus when receiving the second I-frame, the secondary station will return the ACK-frame [<AK,1>], which is supposed to be correctly received. P will in this case retransmit the I-frame [<DT,0,a>], which is supposed to be correctly received by S. Finally S will return the ACK-frame [<AK,0>], which is supposed to be correctly received by P. Suppose for example that the send and receive window size is equal to 2 and that we begin in the following state:

\[\text{FROM\_USER,\_a} \otimes (\text{b}) \otimes (\text{VS,0}) \otimes (\text{P\_T\_O,\_to}) \otimes (\text{P\_WINDOW\_EDGE,\{<0,2>,<2,4>\}}) \otimes (\text{R\_LIST,\_\_\_}) \otimes (\text{TIMER,\_\_\_}) \otimes (\text{DT\_TO\_SEND,\_\_\_}) \otimes (\text{MSG\_O,\_\_\_}) \otimes (\text{ACK\_O,\_\_\_}) \otimes (\text{ACK\_EXPECTED,\_\_\_}) \otimes (\text{LIST,\_\_\_}) \otimes (\text{S\_REC\_COUNT,\_\_\_}) \otimes (\text{W\_SIZE,\{2\}}) \otimes (\text{TO\_USER,\_\_\_}) \otimes (\text{REC\_LIST,\_\_\_})\]

Thus by consequently firing DT\_FRAMING, DT\_FRAMING concurrently with DT\_SENDING, DT\_SENDING concurrently with LOSE\_MSG, DT\_EXPECTED, ACK\_EXPECTED, RETRANS, DT\_EXPECTED concurrently with ACKING, ACK\_EXPECTED concurrently with S\_WINDOW\_EDGE\_UPDATE, ACKING concurrently with P\_WINDOW\_EDGE\_UPDATE, we obtain the following final state:

\[\text{FROM\_USER,\_\_\_} \otimes (\text{VS,2}) \otimes (\text{P\_T\_O,\_\_\_}) \otimes (\text{P\_WINDOW\_EDGE,\{<2,4>,<1,2>\}}) \otimes (\text{R\_LIST,\_\_\_}) \otimes (\text{TIMER,\_\_\_}) \otimes (\text{DT\_TO\_SEND,\_\_\_}) \otimes (\text{MSG\_O,\_\_\_}) \otimes (\text{ACK\_O,\_\_\_}) \otimes (\text{ACK\_EXPECTED,\_\_\_}) \otimes (\text{LIST,\_\_\_}) \otimes (\text{S\_REC\_COUNT,\_\_\_}) \otimes (\text{S\_WINDOW\_EDGE,\{<2,4>,<1,2>\}}) \otimes (\text{W\_SIZE,\{2\}}) \otimes (\text{TO\_USER,\_\_\_}) \otimes (\text{REC\_LIST,\_\_\_})\]

CONCLUSION

In this paper we showed through a case study how to use ECATNets for modeling in a compact and modular way the different actions of an OSI data-link layer protocol based on the selective retransmission strategy. By the way we showed how the different actions (modular blocks) may be rapidly prototyped by giving the corresponding rewrite rules. Moreover we showed how the prototype of the entire protocol may be executed with a maximum of true concurrency according to the defined semantic framework. It is worthwhile to mention that the prototype of the entire protocol is obtained just by putting together the prototypes of its actions (i.e. building blocks). In section 4 we have shown how executions of such a prototype contribute to check the designed model against the expected one, and thus to make safety verifications. Let us notice that the different building blocks identified in this study are not totally dedicated to the data-link layer protocol, but they may be reused at the vertical level. For instance in the different classes of the transport layer merely by dissociating the flow control from the error one. These building blocks may also be used at the horizontal level, i.e. in a data-link layer protocol based on the GO-Back-N retransmission strategy. It is worthwhile to mention that different studies have been realized in this area and submitted for publications. Our support tools consist in an open environment integrating mainly a graphical editor-simulator running on a SUN workstation (Bounouioua and Bettaz, 1992), and a semi-graphical language to build around OBJ-C (Zeghib, 1992). We terminate this paper by mentioning that other works related to the applicability of ECATNets are mainly to the areas of hardware diagnosis and fast protocols are in preparation.

REFERENCES


