PROTOCOL PROJECTIONS: A METHOD FOR ANALYZING COMMUNICATION PROTOCOLS*

Simon S. Lam and A. Udaya Shankar

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

Abstract

We propose the method of protocol projections to facilitate the analysis of protocols with several distinguishable functions. Image protocols are constructed separately for each protocol function. Image protocols are obtained by aggregating protocol entity states, messages and transitions using equivalence relations. As a result, image protocols are typically much simpler than the original protocol and can be analyzed more easily using available techniques. We have shown that if an image protocol satisfies a well-formed property, then it is faithful, in the sense that its logical correctness properties are the same as the logical correctness properties of the projected function in the original protocol. In this respect, the well-formed property can be regarded as a criterion for well-structured protocols.

1. INTRODUCTION

A communication protocol defines the set of rules that govern the exchange of messages between protocol entities. Such protocol entities are connected by communication channels (real or virtual). Keller [1] has described an abstract model for representing parallel computations. The model is a transition system specified by the pair (G,T) where G is the set of global states and T is a binary relation in G called the set of transitions. Models of communication protocols may be represented by such a transition system. The global state of a model of interacting protocol entities is specified by a joint description of the states of the protocol entities and communication channels. State transitions correspond to the occurrence of various events: an entity sending or receiving messages, errors in channels, an entity receiving signals from its user and timers, etc.

Given an initial state g_0, T determines the reachability tree (space) R. R is a directed graph with nodes being elements of G and arcs being elements of T. R contains all available information on logical correctness properties of the protocol. Let R_s denote the set of reachable states in G. R_s, which may be obtained from R, determines the safety (partial correctness) properties of the protocol. Liveness properties, however, require knowledge of the set of finite paths in R.

* This work was supported by National Science Foundation Grant No. ECS78-01803.

Protocol Analysis Approaches

There are three basic approaches to protocol analysis [2]. They differ mainly in the kind of questions that the analyst poses in the characterization of R.

(1) Reachability analysis

Starting from the initial state g_0 in G, the reachability tree R is traversed using T. Many logical errors (e.g., state deadlocks, unspecified receptions, etc.) that require only knowledge of individual reachable states in R can be detected by automated procedures [3]. This approach has two difficulties. First, for any nontrivial protocol, R is extremely large (possibly infinite). Second, many meaningful relationships among "state variables", expressing desirable logical correctness properties of the protocol, require an overall view of R and these are usually not obvious from simply traversing R.

(2) Proofs of invariant assertions

In this approach [4,5], one attempts to formulate invariant assertions describing desired logical correctness properties (corresponding, hopefully, to the service specifications of the protocol). Note that assertions are predicates on G. Hence an assertion can be viewed as a set J of states in G for which the predicate is true. An assertion of a safety property is invariant if the set J is a superset of R_s. Invariant assertions of liveness properties can be thought of as specifying target sets in G for trajectories of paths in R to hit, eventually, recurrently, etc. In any case, the method of formulating invariant assertions, in effect, attempts to find "bounds" for the reachability tree R instead of traversing it. These bounds are in the form of predicates which are statements about the state variables and their relationships.

(3) Hybrid approach

Both of the above analysis approaches may be used together [1,6,7]. For many models, the state space G can be written as a cross product of a set G_c of control states and a set G_d of data states, $G = G_c \times G_d$. A reachability analysis in G_c yields *the reachable* set of control states. For each reachable control state, assertions may be formulated as predicates on G_d.

The difficulty with both the second and third approaches is that the formulation of assertions stating some desired logical correctness properties requires considerable human ingenuity, and cannot be easily automated. However, given some assertions (perhaps derived from protocol service specifications), their verification may be facilitated by various semi-automatic systems using theorem proving and symbolic execution techniques [7, 8].

Analysis of Nontrivial Protocols

The alternating bit protocol has been analyzed by many authors using a variety of techniques [6, 7, 9-12]. This is a relatively simple protocol whose reachability tree R can be easily determined. The X.21 protocol has also been analyzed using automated systems for reachability analysis [13, 14].

The analysis of a relatively large protocol such as HDLC [15] is much more difficult. Reachability analysis typically cannot be employed due to the many state variables, some of which such as sequence numbers can take on a large number of values. Brand and Joyner [16] analyzed one version of HDLC using a system for symbolic execution. Altogether 1347 theorems had to be proved, 80% of which were proved automatically. The rest was done manually.

A protocol, such as HDLC, can be thought of as having many distinct functions. For instance, we can think of an HDLC protocol entity as consisting of five functional components such as shown in Fig. 1. Each component communicates with a corresponding component in the other protocol entity to accomplish a specific function (e.g., connection management, one-way data transfer, etc.).

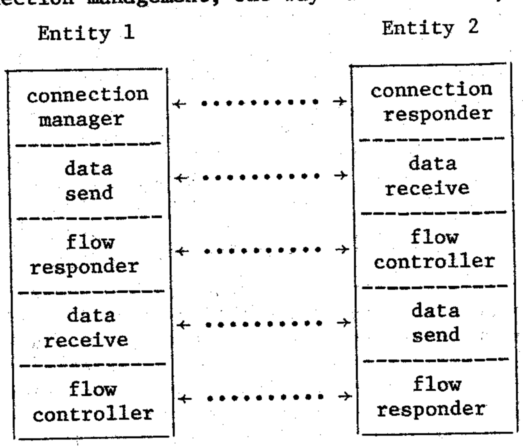

Fig. 1. Functional components of HDLC.

For such complex protocols, an approach that appears attractive is to decompose each protocol entity into modules for handling the different functions of the protocol. Bochmann and Chung [17] used this approach to specify a version of HDLC. The main difficulty with protocol analysis using a decomposition approach is that the interaction among modules within each protocol entity is not structured. Modules interact through shared variables. They also interact because a message frame (packet) typically encodes data and control messages sent by different modules in one entity to the respective modules in the other entity. To model such dependencies, Bochmann and Chung [17] proposed the use of coupled transitions between modules. This permits them to specify the HDLC protocol in a compact fashion, but does not seem to facilitate analysis of the protocol.

The data transfer protocol of Stenning has been successfully analyzed [4, 5]. Invariant assertions concerning both its safety and liveness properties have been proved. However, Stenning's protocol is a one-way data transfer protocol. It corresponds to the interaction of a Data Send module and a Data Receive module in isolation (see Fig. 1.). As such, it constitutes just one function of a real-life protocol such as HDLC. The following question arises: are the safety and liveness properties that are proved for the one-way data transfer protocol still valid when it is implemented as part of an overall protocol with the two types of dependencies mentioned above?

Protocol Analysis via Protocol Projections

Consider a protocol with several distinguishable functions. We would like to ask questions regarding the logical correctness behavior of the protocol concerning these functions. Instead of asking such questions all at the same time, we may ask them with respect to one function of the protocol at a time. The analysis approach is to construct from the given protocol an image protocol for each of the functions that are of interest to us. These will be referred to as the projected functions. The image entity states, messages, and transitions of an image protocol are obtained from those of the original protocol using certain equivalence relations. Entity states, messages and transitions that are equivalent with respect to a projected function are aggregated in the image protocol. As a result, image protocols are simpler than the original protocol. Each image protocol can thus be more easily analyzed using available means. (An image protocol for the function of one-way data transfer of HDLC will be of the same complexity as Stenning's data transfer protocol and can thus be analyzed.) Thus the method of protocol projections is useful if image protocols are faithful, i.e. logical correctness properties of an image protocol are the same as logical correctness properties of the projected function in the original protocol.

A sufficient condition for an image protocol to be faithful is that the image protocol possesses a well-formed property. Although the well-formed property is a sufficient condition, we have found that it is the weakest condition that one can have without any knowledge of the reachability tree R of the protocol. (Note that we cannot assume any knowledge of R since its complexity is the basic source of our difficulties.) We have found that image protocols for a version of the HDLC protocol with respect to the functions of connection management and one-way data transfers possess the well-formed property [18]. Thus the well-formed property is not a very stringent requirement. In fact, one can think of the well-formed property as a criterion of well-structured protocols.

The term "protocol projection" has been previously used by Bochmann and Merlin [19] to describe an operation in their method for protocol construction. Their basic idea of "projection onto the relevant actions" is similar to ours herein, but the development and application of the idea in their work and ours are different.

In Section 2 we shall first present our protocol model. In Section 3 we shall present definitions of image entity states, image messages and image transitions for a given projected function. In Section 4 the image protocol for a given function is defined. The well-formed property is introduced. Our main results are stated in two theorems. We shall illustrate the protocol model with a small protocol example. A small example is chosen for the sake of clarity, although the method of protocol projections is intended for the analysis of real-life protocols with multiple functions.

2. THE PROTOCOL MODEL

Our model is based upon the model of Brand and Zafiropulo [20]. Let P_1 and P_2 be two protocol entities that communicate with each other. P_1 sends messages to P_2 through channel C_1 and P_2 sends messages to P_1 through channel C_2. (See Fig. 2.) Messages transmitted through a channel may be corrupted by noise. The channels are modeled as FIFO queues with infinite storage capacities.

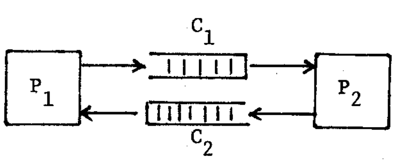

Fig. 2. Components of the protocol model.

(Extension of results in this paper to models of channels that may duplicate and reorder messages and channels that have finite storage capacities will be presented in a forthcoming technical report and in [18].) For i = 1 and 2, we define the following:

S_i is the set of states of entity P_i,

o_i is the initial state of P_i in S_i, and

M_i is the set of messages sent by P_i.

To simplify our notation, we assume that

$S_1 \cap S_2 = \emptyset$ and $M_1 \cap M_2 = \emptyset$

where $\emptyset$ denotes the null set.

Entity P_i is initially in state o_i. A state transition of P_i from state s to state r due to the occurrence of event e is denoted by the 3-tuple <s, r, e>. There are 3 types of events that cause state transitions to take place in S_i. We describe these events for entity P_1.

(1) <s, r, m>, where $m \in M_1$, denotes a transition in S_1 due to the sending of a message m by P_1 into channel C_1. The set of such "send transitions" is a subset of $S_1 \times S_1 \times M_1$.

(2) <s, r, m>, where $m \in M_2$, denotes a transition in S_1 due to the reception of message m by P_1 from channel C_2. The set of such "receive transitions" is a subset of $S_1 \times S_1 \times M_2$.

(3) <s, r, α>, where α is a special symbol indicating the absence of a message, denotes a transition in S_1 that does not involve a message. These transitions are called <u>interface transitions</u> because the state change is caused by events generated at the entity's local interfaces by its user and timers. The set of interface transitions is a subset of $S_1 \times S_1 \times \{\alpha\}$.

The three types of events for entity P_2 are similarly defined, but with the send transitions in $S_2 \times S_2 \times M_2$, receive transitions in $S_2 \times S_2 \times M_1$, and interface transitions in $S_2 \times S_2 \times \{\alpha\}$. Note that both send and receive transitions affect the states of channels as well as the entities involved. Interface transitions do not affect the states of channels. Let T_i denote the set of state transitions of P_i for i = 1 and 2.

The protocol between P_1 and P_2 is defined by specifying,

$$(S_1, S_2, o_1, o_2, M_1, M_2, T_1, T_2).$$

Let $\underline{m}_i$ denote a FIFO sequence of messages representing the state of channel C_i for i = 1 and 2. There are additional events that cause transitions in the states of the channels. First, each message in a channel may be corrupted by noise (an "error event"). We use the symbol η to represent the corrupted message in $\underline{m}_i$ following the occurrence of an error event. Second, if the first element of $\underline{m}_i$ is a corrupted message, it is received by the destination entity and discarded (an "error detection event"). We assume that both types of events do not affect the states of P_1 and P_2. Define

$$M_i^1 = M_i \cup \{\eta\}$$

and

$$M_i^k = M_i^{k-1} \times M_i^1 \qquad \text{for } k = 2, 3, 4, \ldots.$$

The set of all possible message sequences in C_i is a subset of

$$\underline{M}_i = (\bigcup_{k=1}^{\infty} M_i^k) \cup \emptyset.$$

where $\emptyset$ denotes the empty set.

The state space of the global model of protocol interaction is

$$G = S_1 \times \underline{M}_1 \times \underline{M}_2 \times S_2.$$

Each global state is a 4-tuple $<s_1, \underline{m}_1, \underline{m}_2, s_2>$ where $s_i \varepsilon S_i$ and $\underline{m}_i \varepsilon \underline{M}_i$ for i = 1 and 2. The initial global state g_0 will be assumed to be $<o_1, \emptyset, \emptyset, o_2>$ in the rest of this paper without any loss of generality.

We shall make the assumption that if multiple events in P_1, P_2, C_1 and C_2 occur simultaneously, then such an occurrence can be represented as a sequence of occurrences of events in some arbitrary order. This is called the arbitration condition [1]. In the present model, the arbitration condition is acceptable if mutual exclusion is enforced among event occurrences in the same protocol entities i.e. the state transitions of protocol entities are implemented as indivisible (atomic) operations.

Let τ_i denote the set of state transitions in G due to the 3 types of events in P_i. Let τ_c denote the set of transitions in G due to the occurrence of error events and error detection events in the channels. Define T to be $\tau_1 \cup \tau_2 \cup \tau_c$.

The transition system (G, T) defines the global model of the interaction between P_1 and P_2. Recall that R denotes the reachability tree from the initial state $g_0 = <o_1, \emptyset, \emptyset, o_2>$ in G using the binary relation T. A path in R is defined to be an ordered sequence of states in G starting from g_0.

An example

We illustrate the above protocol model with an example of a symmetric full-duplex data transfer protocol to be described next. Since the protocol is full-duplex, it has two basic distinguishable functions: one-way data transfer from P_1 to P_2 and one-way data transfer from P_2 to P_1. For each direction, the protocol uses a very simple acknowledgement scheme. For simplicity, we have assumed that the channels used in this example are error-free. (In this case the protocol does not need timers).

Consider protocol entity P_1. P_1 has an infinite array of data blocks, SOURCE[i] for i = 0,1, 2,..., destined for P_2, and an infinite empty array, SINK[i] for i = 0,1,2,..., to store data blocks received from P_2. The state of P_1 is specified by the values of the 5-tuples

<VS, D_OUT, VR, ACK_DUE, BUSY>

where VS and VR are nonnegative integers while the others are boolean variables. Entity P_2 has a similar set of variables. For convenience, we have omitted qualifiers (1 or 2) for these variables and we shall omit them as long as it is clear whether we are referring to P_1 or P_2. The initial state of both entities is <0, false, 0, false, false>.

The set M_1 of messages sent by P_1 is {DATA1, ACK1, DATA1&ACK1} where DATA1 is a data block, ACK1 is an acknowledgement and DATA1&ACK1 is a message encoding both DATA1 and ACK1. The set M_2 is {DATA2, ACK2, DATA2&ACK2}.

The set of events that cause state transitions in P_1 is presented in Table 1. Events 1-3 correspond to the sending of a message by P_1. Events 6-8 correspond to the reception of a message by P_1. Events 4 and 5 correspond to interface transitions caused by an external agent locally connected to P_1 (e.g., user, channel controller). The enabling condition of an event defines the set of entity states at which the event may take place. The action of each event causes the entity to enter a new state. Thus, each entry in Table 1 actually specifies a set of transitions. For example, the event SEND_DATA defines the set of transitions {<s, r, DATA1> : DATA1 ε M_1, s and r are 5-tuples in S_1 such that s satisfies D_OUT = BUSY = false and r is the same as s except that D_OUT is true and VS is incremented by 1}.

In Table 1, the operation put(CHANNEL, DATA&ACK) encodes DATA and ACK into a single message and appends it to the end of the sequence of messages in CHANNEL. The operation get(CHANNEL, DATA&ACK) removes the message DATA&ACK at the head of CHANNEL and decodes it into two messages DATA and ACK.

The transitions of P_2 are identical to those shown in Table 1 for P_1, except that the variables CHANNEL1, DATA1 and ACK1 and the variables CHANNEL2, DATA2 and ACK2 are interchanged.

The example protocol has two functions corresponding to data transfers in the two directions. The protocol is extremely simple but it embodies the two types of dependencies that one encounters when one attempts to decompose protocols into functional components. First, the variable BUSY is shared by both functions of the protocol. Second, the messages DATA1&ACK1 and DATA2&ACK2 are also shared. Such dependencies present major obstacles for protocol analysis using a decomposition approach. However, the method of protocol projections will be used to obtain a faithful image protocol for each function.

3. PROJECTIONS

Projections are achieved using equivalence relations. We shall denote the image of a protocol quantity x by x' which is obtained by aggregating all those protocol quantities that are equivalent. Images of protocol entity states, messages and global states are next defined and should be obtained in the order shown below.

Projection of entity states

We start by partitioning the state space S_i of protocol entity P_i for i = 1 and 2. Partition cells are defined so that entity states within a partition cell are equivalent with respect to the protocol function for which an image protocol is desired. The equivalence relation is determined by the semantics of the protocol and this is one place in our method where human ingenuity is needed. Let S_i' denote a partition of S_i. S_i' constitutes the set of image states of the protocol entity P_i. Each element in S_i' corresponds to a partition cell in S_i. Thus, if the image of $s \varepsilon S_i$ is $s' \varepsilon S_i'$ then $s \varepsilon s' \subseteq S_i$. We say that states s and r have the same image if and only if both s and r are in the same partition cell (they are equivalent with respect to the projected function).

Projection of messages

The above equivalence relation is next extended to messages in M_1 and M_2. Two messages m and n in M_i are said to be equivalent if and only if they cause identical transitions in both image state spaces S_1' and S_2'. In other words, $m' = n'$ if and only if for any pair of states (s', r') in S_1' and in S_2', m causes a transition from some state in s' to some state in r' whenever n causes a transition from some state in s' to some state in r'. Messages that cause only transitions internal to image entity states in both S_1' and S_2' are said to have the null image. The image message sets are defined by

$M_i' = \{m': m' \text{ is not null}, m \varepsilon M_i\}$.

The image of a corrupted message η is the null image. The image $\underline{m}_i'$ of channel state $\underline{m}_i$ is obtained by taking the image of each element in $\underline{m}_i$ and deleting the null images. Thus for the projected function, transitions associated with null images behave just like interface transitions that are internal to image entity states. (Note that the previous statement is not true when channels with finite storage capacities are considered.)

Images of global states

The above equivalence relations can now be used to define image global states. Two global states $g = \langle s_1, \underline{m}_1, \underline{m}_2, s_2 \rangle$ and $h = \langle r_1, \underline{n}_1, \underline{n}_2, r_2 \rangle$ in G are equivalent ($g' = h'$) if and only if $s_i' = r_i'$ and $\underline{m}_i' = \underline{n}_i'$ for i = 1 and 2. The image global state space is denoted by G'. Let R_s' be the set of images of the reachable global states in R. The image w' of a path w in R is given by the sequence of image global states in the path w where consecutive elements with the same image are aggregated into a single element (in w').

4. IMAGE PROTOCOL

Our objective is to define an image protocol that is faithful, i.e., such that its logical correctness properties are the same as the logical correctness properties of the projected function in the original protocol. A natural candidate for such a protocol is to take S_1' and S_2' as its entity state spaces with o_1' and o_2' as initial states and M_1' and M_2' as its message sets. The transitions T_1' at P_1 and T_2' at P_2 are defined as follows

$$T_i' = \{\langle s', r', m' \rangle : \langle s, r, m \rangle \varepsilon T_i \text{ such that } ((m' \neq \text{null image}) \wedge (m' = \alpha \Rightarrow s' \neq r'))\}$$

where parallel transitions from the set s' to the set r' in T_i associated with messages with the same non-null image are aggregated into a single image transition in T_i'; parallel interface transitions from s' to r' are also aggregated into a single image interface transition (s', r', α) where the image of α is still α.

Note that interface transitions in T_i between states within the same partition set do not contribute to T_i'. Similarly, transitions in T_i involving messages with the null image also do not contribute to T_i'.

Finally, transitions in the channels (due to the occurrence of error events and error detection events) involving messages with the same image are aggregated into a single image channel transition.

The image protocol is completely specified by $(S_1', S_2', o_1', o_2', M_1', M_2', T_1', T_2')$.

Define $T' = \tau_1' \cup \tau_2' \cup \tau_c'$ where τ_1', τ_2' and τ_c' are the global transitions due to respectively T_1', T_2' and image channel transitions. Given (G', T'), let R" be the reachability tree of the image protocol and R_s'' be the set of reachable states in R".

Theorem 1. (i) The image of every path in R is a path in R". (ii) $R_s' \subseteq R_s''$.

A proof of the above theorem is given in [21] and is omitted here due to space limitation. We note that since invariant assertions of the safety properties of the image protocol correspond to supersets in G' that cover R_s'', the second result in Theorem 1 implies that such assertions are also valid in the original protocol. However, it is possible that assertions concerning the projected function are true in the original protocol but are not true in the image protocol. Also, we cannot draw any conclusions about the liveness properties of the projected function in the original protocol based upon the image protocol.

We present next sufficient conditions for the image protocol to be faithful to the projected function. The following axiom is first stated.

Channel error axiom. If channel C_i corrupts messages, then any message in M_i may be corrupted.

Next consider $t' = (s', r', m') \varepsilon T_i'$ and some entity state $a \varepsilon s'$. r' is immediately m'-reachable from a if either: (i) for t' being an interface or send transition, there exists $(a, b, n) \varepsilon T_i$ for some $b \varepsilon r'$ and some n where $n'=m'$, or (ii) for t' being a receive transition, there exists $(a, b, n) \varepsilon T_i$ for all n such that $n'=m'$ and for some $b \varepsilon r'$. r' is m'-reachable from a if there exists a sequence of interface transitions internal to s' that will take a to another entity state $c \varepsilon s'$ from which r' is immediately m'-reachable. (In the protocol model of interest in this paper, send transitions in T_i involving messages with null images can be regarded as internal interface transitions for the above definition).

Definition. Partitions S_1' and S_2' of an image protocol are said to be well-formed if for any $(s', r', m') \varepsilon T_1' \cup T_2'$ and for any $a \varepsilon s'$, r' is m'-reachable from a.

Definition. Partitions S_1' and S_2' of an image protocol are said to be strongly well-formed if for any $(s', r', m') \varepsilon T_1' \cup T_2'$ and for any $a \varepsilon s'$, r' is immediately m'-reachable from a.

Obviously, an image protocol that is strongly well-formed must also be well-formed. In the above definitions, the sets T_i' contain both interface transitions and transitions involving messages. Also, s' may be the same as r' if m' is the nonnull image of a message.

Given a global state g, we let t(g) denote the global state that results upon executing an enabled transition t, where $t \varepsilon T$.

A path $w = g_0 \rightarrow g_1 \rightarrow \dots \rightarrow g_n$ in R is extendable to a path $x = g_0 \rightarrow g_1 \rightarrow \dots \rightarrow g_n \rightarrow g_{n+1} \rightarrow \dots g_{n+m}$ in R if there exist transitions $t_1, t_2, \dots, t_m$ in T such that $t_i(g_{n+i-1}) = g_{n+i}$. The extendability of paths is defined similarly in R".

Theorem 2. Given a well-formed image protocol, (i) for any path w in R and u' in R", if w'=u' and u' is extendable to v' then w is extendable to x such that x'=v'; (ii) $R_s' = R_s''$.

A proof of the above theorem is given in [21]. Part (i) of Theorem 2 together with part (i) in Theorem 1 imply that the liveness properties of a well-formed image protocol are the same as the liveness properties of the projected function in the original protocol. Since $R_s' = R_s''$ in part (ii) of Theorem 2, the safety properties are also the same. Hence a well-formed image protocol is faithful.

An image protocol of the example

Reconsider the full-duplex data transfer example introduced earlier in Section 2 and illustrated in Table 1. The protocol has two functions, namely, data transfer in the two directions between P_1 and P_2. We shall next present an image protocol of the function of data transfer from P_1 to P_2. Observe that VR and ACK_DUE in P_1, and VS and D_OUT in P_2 are not needed for the P_1 to P_2 data transfer. We let S_1' consist of all 3-tuples of the form <VS, D_OUT, BUSY> and S_2' consist of all 3-tuples of the form <VR, ACK_DUE, BUSY>. In other words, the image of <VS, D_OUT, VR, ACK_DUE, BUSY> in S_1 is <VS, D_OUT, BUSY> in S_1'.

Using the definition for images of messages in Section 3, we find that the image message sets are

$M_1' = \{DATA1'\}$ and $M_2' = \{ACK2'\}$.

Consider M_1'. DATA1' is the image of DATA1 and DATA1&ACK1 in M_1. ACK1 in M_1 has the null image and is not included in M_1'.

Using the definition for image transitions of protocol entities, the set of image transitions of P_1 is found and shown in Table 2, and the set of image transitions of P_2 is found and shown in Table 3.

The image protocol for the function of data transfer from P_1 to P_2 is relatively simple and the following invariant assertions of its logical behavior have been proved.

Invariant assertions

1. SINK[i] = SOURCE[i] for $0 \leq i < VR$.
2. $VS \geq VR \geq VS-1$.
3. DATA1' in CHANNEL1 ⇒ (D_OUT)
   ^ (DATA1' = SOURCE[VS-1])
   ^ (exactly one DATA1' message in CHANNEL1)
   ^ (not ACK_DUE) ^ (VS = VR + 1)
   ^ (no ACK2' message in CHANNEL2).
4. ACK_DUE ⇒ (D_OUT)
   ^ (no DATA1' message in CHANNEL1)
   ^ (VS=VR)^(no ACK2' message in CHANNEL2).
5. ACK2' in CHANNEL2 ⇒ (D_OUT)
   ^ (no DATA1' message in CHANNEL1)^(VS=VR)
   ^ (not ACK_DUE)
   ^ (exactly one ACK2' message in CHANNEL2).
6. not D_OUT ⇒ VS = VR.

It can be shown using the definition of well-formed partitions that the image protocol defined above is strongly well-formed. Therefore, the above invariant assertions also describe the logical behavior of data transfer from P_1 to P_2 in the original protocol.

5. DISCUSSIONS

The method of protocol projections is intended to facilitate the analysis of protocols with several distinguishable functions. Image protocols are defined separately for each protocol function. We have shown that if an image protocol is well-formed, then it is faithful, in the sense that its logical correctness properties are the same as the logical correctness properties of the projected function in the original protocol.

Unlike a decomposition approach, the method of protocol projections is not handicapped by dependencies that exist between different functional components of a protocol due to (1) the sharing of variables, and (2) the encoding of messages for the different functions into the same message frames (packets). Such dependencies are naturally accounted for in the definition of image protocols, since all the entity states and messages that are relevant to a function are included in its image protocol. On the other hand, since image protocols are obtained by the aggregation of equivalent states and messages, they are typically much simpler than the original protocol, and can be more easily analyzed by available verification techniques.

Note that the well-formed property is a sufficient condition for an image protocol to be faithful. However, a careful examination of the proof of Theorem 2 in [21] will show that this is the weakest sufficient condition that one can state without any knowledge of the reachability tree R. We have found that image protocols are well-formed for a version of the HDLC protocol with respect to the functions of connection management and one-way data transfers [18]. Thus, the well-formed property is not as stringent as it may appear.

One interpretation of a well-formed image protocol is that the aggregation of states in S_1 and S_2, to form S_1' and S_2', does not result in any loss of information for the projected function. If S_1' and S_2' were not well-formed, then we have made the error of considering two entity states s and r in S_i as equivalent witn respect to the projected function, though in fact thay are not. In this respect, one can think of the well-formed property as a criterion of well-structured protocols. That is, a protocol would be considered well-structured, if to each of the basic protocol functions, we can define "maximal" partitions S_1', S_2' to obtain image protocols.

We have extended the results in this paper to models of channels that may duplicate and reorder messages. We are also investigating issues of time variables, implementation variables and protocol synthesis, as well as the development of efficient algorithms for obtaining protocol images and checking the well-formed property [18].

REFERENCES

[1] Keller, R.M., "Formal Verification of Parallel Programs," Comm. ACM, July 1976.

[2] Bochmann, G.V. and C.A. Sunshine, "Formal methods in communication protocol design," IEEE Trans. on Commun., April 1980.

[3] Zafiropulo, P., et.al., "Towards Analyzing and Synthesizing Protocols," IEEE Trans. on Commun., April 1980.

[4] Stenning, N.V., "A Data Transfer Protocol," Computer Networks, September 1976.

[5] Hailpern, B.T. and S.S. Owicki, "Verifying Network Protocols using Temporal Logic," Computer Systems Laboratories, Stanford Univ. Technical Report No. 192, June 1980.

[6] Bochmann, G.V. and J. Gecsei, "A Unified Method for the Specification and Verification of Protocols," IFIP Congress 77, North Holland Publishing Company, 1977.

[7] Brand, D. and W.H. Joyner, "Verification of Protocols using Symbolic Execution," Computer Networks, September/October 1978.

[8] Good, D.I. and R.M. Cohen, "Principles of Proving Concurrent Programs in Gypsy," Proc. 6th Annual ACM Symposium on Principles of Programming Languages, San Antonio, Jan. 1979.

[9] Bochmann, G.V., "Finite State Description of Communication Protocols," Computer Networks, October 1978.

[10] Merlin, P.M., "Specification and Validation of Protocols," IEEE Trans. Commun., Nov. 1979.

[11] Bremer, J. and O. Drobnik, "A New Approach to Protocol Design and Validation," IBM Research Report RC8018, Yorktown Heights, New York, December 1979.

[12] DiVito, B.L., "A Mechanical Verification of the Alternating Bit Protocol," University of Texas at Austin, Technical Report ICSCA-CMP-21, June 1981.

[13] West, C.H. and P. Zafiropulo, "Automated Validation of a Communications Protocol: The CCITT X.21 Recommendation," IBM Journal of Res. and Develop., January 1978.

[14] Razouk, R. and G. Estrin, "Modeling and Verification of Communication Protocols in SARA: The X.21 Interface," IEEE Trans. on Comput., December 1980.

[15] International Standards Organization, "Data Communication--HDLC Procedures--Elements of Procedures," Ref. No. ISO 4335, 1979.

[16] Brand, D. and W.H. Joyner, "Verification of HDLC," IBM Research Report RC7779, Yorktown Heights, New York, July 1979.

[17] Bochmann, G.V. and R.J. Chung, "A Formalized Specification of HDLC Classes of Procedures," Proc. Nat. Telecommun. Conf., Los Angeles, December 1977.

[18] Shankar, A.U., "Analysis of Communication Protocols via Protocol Projections," Ph.D. Thesis, Dept. of Electrical Engineering, Univ. of Texas at Austin, 1982 (in preparation).

[19] Bochmann, G.V. and P. Merlin, "On the Construction of Communication Protocols," Proc. 5th ICCC, Atlanta, October 1980.

[20] Brand, D. and P. Zafiropulo, "On Communicating Finite-State Machines," IBM Res. Report, RZ1053, Zurich, Switzerland, Jan. 1981.

[21] Lam, S.S. and A.U. Shankar, "Protocol Verification via Protocol Projections, Part 1: Theory," Dept. of Computer Sciences, Univ. of Texas at Austin, Technical Report, 1981 (in preparation).

Event Name	Enabling Condition	Action
1. SEND_DATA	not BUSY and not D_OUT	DATA1 := SOURCE[VS]; put(CHANNEL1, DATA1); VS := VS + 1; D_OUT := true
2. SEND_DATA_ACK	not BUSY and not D_OUT and ACK_DUE	DATA1 := SOURCE[VS]; put(CHANNEL1, DATA1&ACK1); VS := VS + 1; D_OUT := true; ACK_DUE := false
3. SEND_ACK	not BUSY and ACK_DUE	put(CHANNEL1, ACK1); ACK_DUE := false
4. START_BUSY	not BUSY	BUSY := true
5. STOP_BUSY	BUSY	BUSY := false
6. REC_DATA	first(CHANNEL2) = DATA2	get(CHANNEL2, DATA2); SINK[VR] := DATA2; VR := VR + 1; ACK_DUE := true
7. REC_DATA_ACK	first(CHANNEL2) = DATA2&ACK2	get(CHANNEL2, DATA2&ACK2): SINK[VR] := DATA2; VR := VR + 1; ACK_DUE := true; D_OUT := false;
8. REC_ACK	first(CHANNEL2) = ACK2	get(CHANNEL2, ACK2); D_OUT := false

TABLE 1. Transitions of Entity P_1 in the Protocol Example.

Event Name	Enabling Condition	Action
1. SEND_DATA'	not BUSY and not D_OUT	DATA1' := SOURCE[VS]; put(CHANNEL1, DATA1'); VS := VS + 1; D_OUT := true
2. START_BUSY	not BUSY	BUSY := true
3. STOP_BUSY	BUSY	BUSY := false
4. REC_ACK'	first(CHANNEL2) = ACK2'	get(CHANNEL2, ACK2'); D_OUT := false

TABLE 2. Transitions of P_1 in the Image Protocol.

Event Name	Enabling Condition	Action
1. REC_DATA'	first(CHANNEL1) = DATA1'	get(CHANNEL1, DATA1'); SINK[VR] := DATA1'; VR := VR + 1; ACK_DUE := true
2. START_BUSY	not BUSY	BUSY := true
3. STOP_BUSY	BUSY	BUSY := false
4. SEND_ACK'	not BUSY and ACK_DUE	put(CHANNEL1, ACK2'); ACK_DUE := false

TABLE 3. Transitions of P_2 in the Image Protocol.