Using Knowledge-based System Techniques in the Protocol Design Process

M. Nawaz Brohi
Department of Information Technology
Preston University
P.O.Box: 20488 Ajman
United Arab Emirates

ABSTRACT
This paper describes the knowledge-based system techniques, such as program transformation and artificial intelligence techniques, used to design a reliable protocol. Program transformation techniques can be used in deriving protocol specifications. AI techniques, such as search algorithms and theorem proving, can be used to reduce the global space search. AI techniques can also be used to help correctness proving in protocol validation and verification. This study is based on the Alternative Bit Protocol (ABP).

Keywords
Axioms, correctness proving, deductive inference, intelligent assistant, knowledge-based system, interface engine, protocol validation, protocol verification, reachability analysis.

1. INTRODUCTION
A knowledge-based system (KBS) is a method of encoding and decoding human expertise into mechanically manipulated form and vice versa. A knowledge-based system is a new way of encoding human expertise into mechanically manipulated forms [1]. It consists of two major elements: knowledge base (KB) and inference engine (IE). The KB is the backbone of expert system, which corresponds to a program in conventional automatic problem-solving systems, is a collection of encoding knowledge expressed in some formal representation. In a rule-based expert system this knowledge is represented in the form of conditional (if … then …) rules. The KB contains both general knowledge as well as case specific information. The inference engine, which corresponds to an interpreter in conventional systems, is a control mechanism to manipulate the representation in the knowledge base. These two components together provide a new reeking of problem solving that deals with the encoding of the human's expertise much better than any standard procedure language. The IE applies the knowledge to the solution of actual problems. It is essentially an interpreter for the KB.

Protocols are the rules defining the communication between two entities. Design of just a simple protocol is not a difficult task but designing of reliable and error free protocol is really a challengeable activity because of the complexity of the rules defining the interaction between the communicating entities. It requires a lot of human expertise to specify, design and implement a correct protocol. In this regard KBS can play very important role in the protocol specification, validation and verification.

1.1 Protocol Designing Rules
During protocol design process we must follow the following basic rules:

a. **Problem Definition:** Make sure that the problem is well defines. All design criteria, requirements and constraints, should be listed before a design is started.
b. **Service Definition:** Define the service to be performed at every level of abstraction before deciding which structures should be used to realize these services.
c. **Functionality:** Design external functionality before internal functionality.
d. **Simplicity:** Keep it simple.
e. **Independency:** Do not connect what is independent.
f. **Immaterial Freedom:** Do not introduce what is immaterial. Do not restrict what is irrelevant.
g. **Prototype Design:** Before implementing a design, build a high-level prototype and verify that the design criteria are met.
h. **Implementation:** Implement the design, measure its performance and if necessary, optimize it.
i. **Verification:** Check that the final optimized implementation is equivalent to the high-level design that was verified.
j. Do not skip Rules **a** to **g**.

2. PROTOCOL VALIDATION
The most important factor for any communication system is the correctness of a protocol. We should make sure that a communication system must be reliable and unambiguous. Make sure that implemented protocol must be correct. To achieve this goal, we must design and implement communication protocols carefully. Verification and Verification process can guarantee the correctness of a protocol. Verification and validation are often used interchangeably. Protocol verification is the process of formally analyzing a protocol specification by defining a set of axioms representing the protocol and formally proving that the specification satisfies the axioms and hence correct [2]. Protocol validation process method is less reliable compared to verification. Protocol verification method is more accurate tool to take care of correctness of a protocol. Analysis is limited to an incomplete set of general protocol properties like deadlocks, livelocks etc.

We will follow the terminology used by Sunshine [3]. That is, protocol verification is a demonstration that the interactions of the communicating entities, based on their protocol specification and the specification of the services provided by the layer below, satisfy the service specification, whereas protocol validation refers to the more limited analysis that the protocol specification satisfies a number of general correctness properties that are essential to all, or nearly all, protocols. The list of general correctness properties that must be satisfied by all protocols is as follows:

- Perfectness and Completeness: The protocol must be able to take care of all the requirements that may be arising by the end-user during the communication.
- Deadlock Independency. States in which no further protocol execution is possible, for instance because all protocol processes are waiting for conditions that can never be fulfilled. The protocol must be free to take care of each state during communication. Each global state allows the other state to progress. Each global state should be reachable.
- Meaningless Loops. Avoid using meaningless looping paths. All looping paths must provide some meaningful communication tasks.
- Livelock Independency. Execution sequences that can be repeated indefinitely often without ever making effective progress. If there is a meaningless loop then it must provide some exit to paths along which meaningful communication tasks may take place.
- Overflow Independency. Communication channels must be greater than the communication messages otherwise we have to use a buffer to avoid the overflow.
- Termination Process. The completion of a protocol execution with satisfying the proper termination conditions. The protocol must reach at final state, called accepted or desired state.

From the above discussion, we learnt that the approaches the protocol validation depends heavily on the models used for specification and they have followed two main paths: reachability analysis and deductive inference. Reachability analysis is based on exhaustively exploring all the possible interactions of the communicating protocol entities from a layer, whereas deductive inference is based on a list of statements of properties and a list of axioms and rules for inferring the statements from the axioms. Restricting our discussion in this section to reachability analysis only, we will study relief strategies proposed by many researchers to deal with the so-called state explosion problem and also propose an approach, which is based on the search strategies developed in the field of artificial intelligence.

3 REACHABILITY ANALYSIS

Reachability analysis has been proved to be one of the most effective ways for analyzing state-oriented models of communication protocols. It was first proposed by West [4] and later improved by a number of researchers. The method is based on the idea of state perturbation in which all the possible global states of a protocol are enumerated from an initial state. Properties of the protocol can then be verified based on the global states and the global state reachability graph.

Validation techniques used by FSA models are all based on some sort of reachability analysis. This analysis involves the exploration of all possible interactions among communicating entities. A global, system or composite state is a combination of both the states of communicating entities and the states of communication media. From the initial global state, new global states are generated by applying all possible transitions (user commands, message arrivals, and internal timeouts). This process continues for each newly generated global state until no new states can be generated. The resulting graph is called the reachability graph.

Reachability analysis is well suited to checking the general correctness properties described above since these properties are a direct consequence of the structure of the reachability graph. For example, global states with no exits are either deadlock states or proper termination states. More importantly the generation of the global state space can be easily automated and several automated systems for protocol validation have been developed. A global state graph of the above ABP is given in Figure 1 and Figure 2.

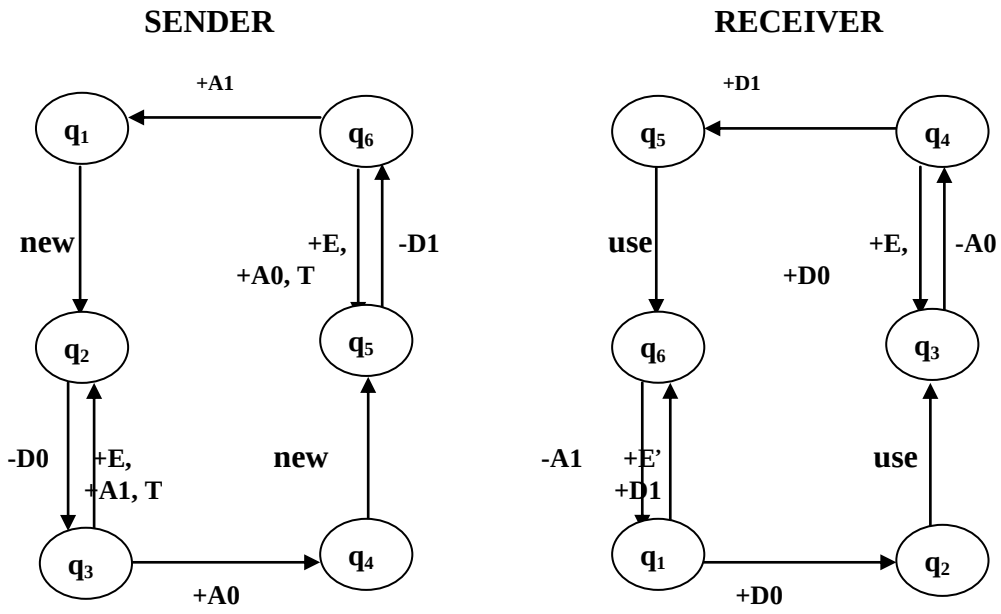

Figure 1: FSA model of the ABP with timeout mechanism

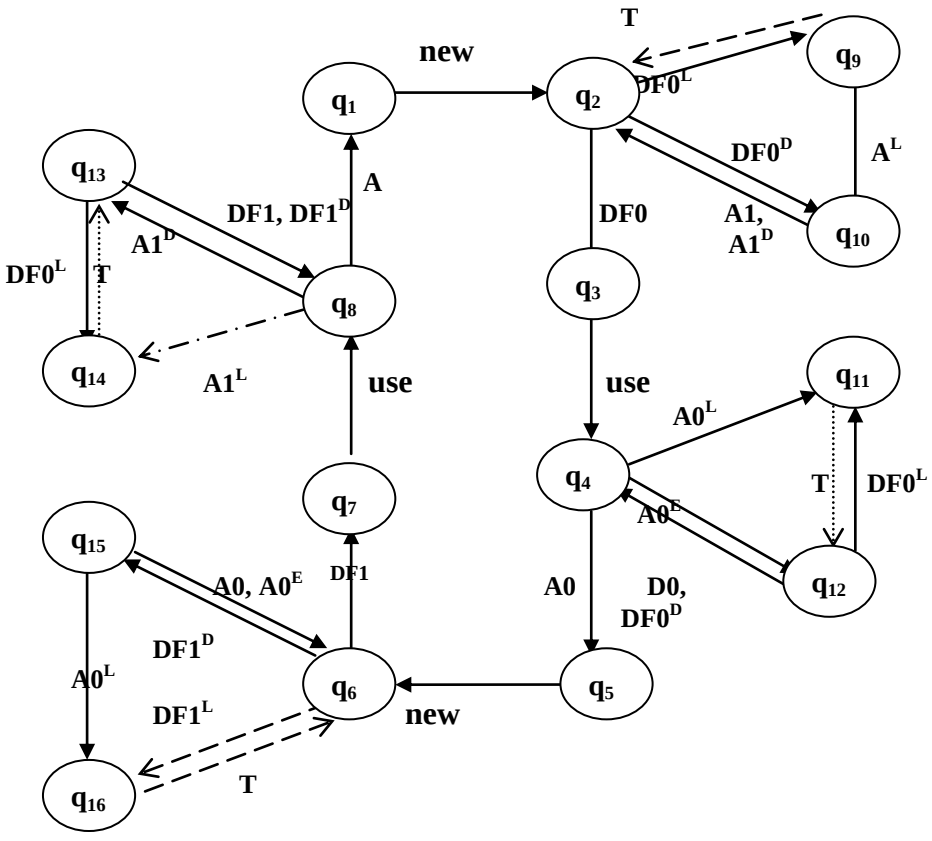

Figure 2: Global State Graph (GSG) of the ABP under the empty medium abstraction

DF^D = Data Frame Damaged on Transmission
DF^L = Data Frame Lost on Transmission

This graph is generated under the assumption of so-called empty medium abstraction. Under this assumption, communication media are considered empty, i.e., no message is in transit. Therefore, a global state is composed of the states of the communication entities.

Mathematical GSG = (S, R), in the GSG, (S, R) represents a global state where the sender is in state S and the receiver is in state R.
A possible transition consists of the sending of a message by sender and the reception of this message (or message with error) by the receiver. In the case of message loss, a transition corresponds to only the sending of message. The transition labeled T, stands for reliable transmission (followed by reception) of a data frame with control bit **i**, where **i** is 0 or 1**.** DFi^D shows that the data frame is damaged on transmission and the frame with error is received by the receiver. DFi^L represents that the data frame is lost on transmission and no data frame is received by the receiver. The same terminology is used for the acknowledgement frame except that DF has been changed to A.

3.1 Alternative Bit Protocol Algorithm

The ABP is used to guarantee the correct data delivery between a sender (S) and receiver (R) connected by an error channel that loses or corrupted messages. It got the name since it uses only one additional control bit in the message and this control bit only alternates when the previous message is correctly received. The sender sends one frame and waits until the status of this frame is determined. If the sender has received a positive acknowledgment (+A) in a predetermined amount of time, the sender sends the next frame. Otherwise, the sender resends the frame.

- The sending device keeps a copy of the last frame transmitted until it receives an acknowledgment for that frame.
- For identification purposes, both data frames and A frames are numbered 0 and 1 alternately. A data 0 frame is acknowledged by an A1 frame, indicating that the receiver has gotten data 0 and is now expecting data 1.
- If an error is discovered in a data frame, a negative acknowledgment (-A) frame is returned. A –A frame tells the sender to retransmit the last frame send.
- The sending device is equipped with a timer. If an expected acknowledgment is not received within an allotted time period, the sender assumes that the last data frame was lost in transit and sends it again.

3.1.1 Working Structure of ABP

Each message is sent over the medium has a single bit added, either a 1 or a 0. These bits are non-informatics bits called control bits, the use of these bits are determine if message is a duplicate, may be resent after + acknowledgement (ACK) was not received after a predetermined time period. These bits are also allows the sender to determine if an acknowledgement just received is a duplicate. Here we have to consider the case where one message was sent, but the acknowledgement was delayed. The sender resends the message and another ACK was sent. Acknowledgement one arrives, so the next message is sent. If the delayed acknowledgement is received, the sender may assume that the message it just sent was received. In ABP protocol, the loss of message is possible because of unreliable medium.

Consider a sender A and receiver B and assume that the no message in transit. The ABP works as follows:

- Each data message sent by A contains an additional protocol bit, 0 or 1.
- When A sends a message, it sends it repeatedly (with its corresponding bit) until receiving an acknowledgment (ACK) from B that contains the same protocol bit as the message being sent.
- When B receives a message, it sends an ACK to A and includes the protocol bit of the message received. The first time the message is received, the protocol delivers the message for processing. Subsequent messages with the same bit are simply acknowledged.
- When A receivers an ACK containing the same bit as the message it is currently transmitting, it stops transmitting that message, flips the protocol bit and repeats the protocol for the next message.

The resulting global state graph may be examined for detecting general correctness properties. For example, in Figure 2, each global state can go back to the initial global state, thus indicating the absence of deadlocks. There exist lops without progress such as the loop consisting of states q_2 and q_{10}, q_4 and q_{12}, q_6 and q_{15} and q_8 and q_{13}. These loops are executed in the case of transmission errors or losses and may be prevented by setting a limit to the number of retransmission times.

The main advantage of FSA model is that reachability exploration can be automated. The process of validation is far too time consuming and error prone if done by hand. It may be possible to carry out validation on simple protocols by hand. As protocols become more and more complex, the effort of manual validation grows beyond human capability. With the help of an automated validation program, tremendous design time consumed by the protocol designer can be saved.

The major difficulty of such models is state space explosion (the size of the global state graph grows exponentially with protocol complexity). For complex protocols (TCP/IP), this technique becomes too complicated for a complete generation and examination of all reachable global states. Thus, state transition approaches are not all suitable for modeling variables that may take on a large number of values. TCP is designed for reliability; this reliable delivery comes with a cost of slow delivery and more complexity.

4. RELIEF STRATEGIES

Due to its efficiency and case of mechanization as discussed above, many protocol validation tools have been built based on the method of reachability analysis. However, the applicability of this method is severely restricted by the so-called state space explosion problem. Many researchers have developed relief strategies to attack the state space explosion problem. We are presenting a brief review of these strategies as below:

The relief strategies presented here can be classified into three categories according to when they should be applied. The strategies in the first category are those applied during protocol modeling, i.e., in the stage of formally specifying protocols. The second category of relief strategies are applied after the protocol modeling is done but before the actual validation is performed. The third category of strategies is those incorporated into the validation algorithms.

1. The relief strategy proposed by West [5] falls into the first category. The major techniques proposed by West are as follows:

- Limiting the use of many valued parameters such as sequence numbers in the specification.
- Limiting the number of messages in progress in message queues.
- Limiting the classes of transmission errors under consideration.

A reliable protocol inserts a sequence number in a data frame; the data frames must be numbered sequentially to control the damages. To avoid the losses we must limit the number of messages, or size of queue must be greater than the number of messages.

2. Through different terms such as decomposition and multi-phase are used by the groups of researchers, the relief strategies they proposed basically follow the same direction [6]-[8]. They observe that certain classes of protocols can be decomposed into components (or multiple phases), which can then be separately verified to ensure the correctness of the original protocol. This reduces the complexity of the verification problem since protocol components are always smaller in the numbers of states and transitions than the original protocol. They are relief strategies of the second category as classified at the beginning of this section.

3. The strategy proposed by Lam and Shanker [9] also belongs to the second category, these relief strategies are applied after the protocol modeling is done but before the actual validation is performed Unlike the strategies of decomposition Lam and Shankar proposed the projection approach, which, instead of partitioning a protocol into multiple phases, constructs from the given protocol an image protocol for each of the functions that is intended to be verified. The states, messages and events of entities in an image protocol are obtained by aggregating groups of states, messages and events of corresponding entities in the original protocol. The resulting protocol is smaller than the original protocol and therefore the complexity of the problem is reduced. If the complexity of the problem is decreases then the speed of the communication system increases. The category of relief strategies belongs to the validation algorithms presented by the different researchers are presented below:

4. Blumer and Sidhu has built as one of the tools in the protocol development, called the Finite State Machine (FSM) analyzer, is based on the model of the extended finite state machine (EFSM) [10]. A mechanism called transition choice rule is provided, which is associated with each of the transitions. The choice rule is a Boolean condition whose value decides whether or not the associated transition of the FSM is to be extended during the reachability analysis.

5. LISE is also a tool based on the model of the extended finite state machine. It can be operated in two modes: validation mode and simulation mode [11]. When the system is operated in validation mode, it fires all the possible transitions in every global state. On the other hand, if the system is in simulation mode, only one transition out of a global state is selected to fire. The simulation mode is adopted whenever it turns out that a complete validation is infeasible due to state explosion. Selection in simulation mode is accomplished in two ways. In the first way, the selection is simply done on a random basis; in the second way, a priority is assigned to each of the transitions and the transition with the highest priority is always the one chosen.

6. One group of third category strategies is based on the fair progress state exploration [12]-[14]. The same category was first proposed by Rudin and West [12], and then extended by other researchers. The idea is to explore only those global states that are reachable, provided that two protocol entities proceed at the same speed. Protocol design errors such as deadlocks and unspecified receptions can still be completely detected though the exploration is not exhaustive. This strategy has some limitations and it only applies to two-entity protocols.

7. The strategy similar to the strategy based on the fair progress state exploration, called the maximal progress state exploration is presented by one group of researchers [15], [16]. The applicability of this strategy is also limited to two-entity protocols. The strategy is that the global states of a two-entity protocol can be generated in two separate explorations, during each of which a different entity is forced to proceed at its maximum speed whenever possible. The state space explored is not exhaustive. Nevertheless, protocol design errors such as deadlocks, unspecified receptions, and channel overflows can still be detected. In addition, this method has another advantage over others in that it can be structured to run as two processes on two processors to further speed up the validation process.

8. The relief strategy proposed by Itoh and Ichikawa is applicable to protocols whose entity FSMs do not contain any cycle that does not pass the initial states [17]. In each global state, the acceptable events of different entities are executed simultaneously to derive the next global state. Moreover, if there is some potentially acceptable event in the current state of an entity E, additional global state derivations by inhibiting the execution of all the acceptable events of E should also be performed. The purpose is to force entity E to wait in order that any of its potentially acceptable events may become executable later. Following this algorithm, only part of the global state graph is explored. The interaction sequences thus explored are called the reduced implementation sequences and are used to verify the protocol against the given requirement specification.

9. The tree group of strategies is called the acyclic form protocol validation strategies [18], [19]. Instead of discovering the global states of a protocol, this strategy grows each entity of the protocol into a tree or an acyclic form. During the growing process, protocol design errors such as unspecified receptions, deadlocks and channel overflows can be detected. The algorithm of this strategy is much more complicated than the traditional global states reachability analysis. However, the validation speed is improved. Generally in more complicate algorithms, the validation speed is low but in this group the validation speed is moderate.

10. The new tool called trace designed by Holzmann, which also works under two states, either as a fast debugging tool or as a slower correctness prover [20], [21]. The main emphasis is that the user can control the scope of each search. When used as a debugging tool. Trace uses a search strategy called scatter search to explore the global states graphs, which basically is a depth-first search guided by some simple heuristics and restricted by a depth limit. Few discoveries proposed by Holzmann are listed below:

- Restrict the amount of non-determinism.
- Assign priorities among concurrent events.
- Limit queue sizes.

- Discard all the states after the depth-first exploration except those that are loop states.
- Keep a limited size of cache for storing global states.
- Minimize the FSM models of protocol entities before verification.

11. The random-walk state exploration strategy is presented by West [22]. From his experience in validating the Open Systems Interconnection / International Organization for Standardization (OSI/ISO) session layer protocol. West observed that the majority of errors detected are found many times in different global states for a complex protocol. This suggests that an analysis of a subset of the reachable global states may be sufficient to identify a significant fraction of errors. The random-walk strategy is thus proposed as a way to partially explore the global states graph, the strategy is as follows:

- If there is any event that may cause message collision when executed, such an event is fired otherwise, arbitrarily choose any event to fire.
- The state exploration is switched out continuously along a single path without backtracking. As a result, none of the previous states needs to be remembered and the cases that have already been explored may be explored again.

12. The relief strategy proposed by Vuong et al. solves a class of problems that the conventional reachability analysis tail reachable global states induced unbounded accumulation of messages in the media. Their proposed strategy is a new global state representation based on which the reachability algorithm developed can generated finite graphs for all non-FIFO and for a certain cases of FIFO protocols even though these protocols may produce an unbounded number of messages in the transmission media [23].

5. PROTOCOL VALIDATION TESTING (PROVAT) STRATEGY

The majority of the relief strategies described in section-4 are informal, utilizing heuristic information. We believe that the problem should be attacked more systematically by borrowing some states from the search strategies developed in the field of AI. Instead of adopting any of the above discussed strategies in the proposed validation tool, we have developed from a new approach. This strategy is called PROVAT for the following two reasons:

- It is a strategy integrated into a validation tool.
- When the tool resorts to the PROVAT strategy, it is performing the task of design testing instead of design validation, since only some of the reachable global states will be explored. The purpose is to show the existence, not the absence of protocol design errors. Compared to general search problems, the search done on the state space of a protocol has the following distinguished characteristics:
  - Rather than searching for a required or satisfactory solution, validation testing searches for protocol design errors of unspecified receptions, deadlock states and channel overflows.
  - The quality of search strategy is judged by the discovered percentage of the total number of errors in a limited amount of time and space. Better strategies will discover higher percentages of errors in the same amount of time and space.
  - When searching into the protocol state space, pruning can be done based on how likely a subtree of states can be exercised by the protocol operation. In case an exhaustive analysis is infeasible, those states that are more frequently exercised by the protocol should validated first.
  - An efficient search will primarily focus on one type of error at a time because the heuristics required in locating different types of errors may contradict each other.

Like the best first search developed in the AI field, an ideal search in the domain of protocol validation is called the error first search. PROVAT is a first attempt to characterize such an error first search.

As pointed out in the previous section, heuristics can be applied at three points in a search process, namely, the points to decide which global states to expand next, to decide which transitions to fire next and to decide which global states to discard. PROVAT is designed to employ heuristics at all three points.

5.1 Explanation

A global state is said to be generated when its data representation is computed from that of its predecessor (parent). When this occurs, the parent state is said to be expanded. A state is fully expanded if all of its successors (children) are generated; otherwise, it is partially expanded. At some point, each generated state has to be inspected to see whether it reveals any of the protocol design errors. A state is called explored if it has been inspected. In addition, during the validation process, the states generated are dynamically partitioned into two sets: CLOSED and OPEN. In the search algorithm of PROVAT, the generated states that are never or partially expanded are placed in OPEN and those that have been fully expanded are moved to CLOSED. Since a state may remain in OPEN for a long time before it is expanded, it is reasonable to explore the state immediately after it is generated.
In the following, the heuristics used by PROVAT are explained. We assume that the only available protocol operations are "send" and "receive".

5.1.1 Heuristics in deciding which global states to expand next

The purpose is to expand those global states in OPEN that are closer to errors. The heuristics are mainly concerned with the status of queues and entities. For each type of protocol errors, a different heuristics is derived.

5.1.1.1 Undetermined Reception

We count the number of queues that satisfy the following two conditions: (a) the queue is non-empty, and (b) its destination entity is willing to receive the message at the head of the queue. Global states the largest number of this type of queue will be expanded first.

5.1.1.2 Deadlock

Examine all the empty queues in a global state and call N_1 the number of empty queues whose destination entities are in receiving states (states without any outgoing "send" transitions) and N_2 the number of empty queues whose destination entities are not. Global states then are scored

according to the weighted sum of N_1 and N_2. The states that receive the highest score will get the first attention.

5.1.1.3 Channel Overflow

Global states are compared based on the length of their longest queue. If a tie occurs, the comparison continues based on the length of the second longest queue. States that win in this contest will be explored first.

5.1.2 Heuristics in deciding which transitions to fire next

The purpose is to perform those actions that are more likely to lead to the error from a selected state. The heuristics are concerned with either action types or queue lengths. Different heuristics are developed for each type of error.

5.1.2.1 Unspecified Reception

We choose a receive operation if that operations is able to receive a message from the shortest queue that contains at least two messages; otherwise, we choose a "send" operation that sends a message to an empty queue. For other operations, consider receive before send. These heuristics tend to sustain the decision made by the heuristics of choosing the next expanded global state.

5.1.2.2 Deadlock

Receive operations are always considered first. Among the "receive" operations, we choose those which extract from the shortest queue.

5.1.2.3 Channel Overflow

Send operations are always considered first. Among sends, those which add to the longest queue have the highest priorities. If there is no send operation, receives that extract messages from the shortest queue are chosen. The heuristics for the above two types of errors are also derived to be compatible with those in deciding which global states to expand next.

5.1.3 Heuristics in deciding which global states to discard

The purpose is to decide which global states should not be generated during the global state expansion, which in fact prunes the sub-tree rooted by any state thus inhibited. To bring in meaningful heuristics in making this decision, we first enhance the original TG model on which the validation tool is based, to include probability specifications [24]. Then a simple method is developed to estimate how likely each of the global states will be reached in terms of probabilities. Similar work was done on CCS for a different purpose [25]. Global states being assigned smaller probabilities are less likely to be reached by the protocol operation. Consequently, if speed is the major concern to the protocol verifier, any one can ask PROVAT to explore only those states with probabilities of reachability higher than a specified threshold.

All the heuristics informally defined above are quantified by the evaluation functions, which play major roles in guiding the reachability analysis.

Another problem left out in the above discussion is when to terminate the partial state exploration. In PROVAT, this is decided by two criteria:

I. Specifying a probability threshold to explore only the states are more likely to be exercised by the protocol. Those
   global states with probabilities of reachability dropped below the threshold value will never be generated.

II. Specifying an upper bound on the number of expansion steps. When a state is expanded, some new or existing states will
   be generated. Each step of generating a state, whether the resulting state is new or already existing is called an expansion
   value, the analysis terminates.

The first criterion is supported by the third kind of heuristics discussed above. On the other hand, the second criterion gives an approximate estimate on how much time will be taken by the analysis. In PROVAT, the first criterion is used to tailor the state space to contain only those paths that are more likely to be exercised by the protocol, and then the second criterion is applied to obtain a desirable response time-knowledge-based protocol validation system (KBPVS).

It is well known that conventional protocol validation based on reachability analysis suffers a great deal from the state explosion problem. Consequently, many variants of reachability analysis have been proposed in the literature to alleviate this problem. We have surveyed and evaluated these variant algorithms. One of the conclusions we reached is that none of the improved algorithms can totally supersede the others or even the conventional, exhaustive reachability analysis itself. In other words, each algorithm including the conventional one has advantages over the others under certain requirements and conditions. Thus we believe that a better protocol validation system should make these algorithms accessible to the protocol designer. This simple means to provide the protocol designer with a box of validation tools that implement various validation algorithms. We call this way of implementing the validation system the tool box idea.

Nevertheless, only providing the protocol designer with a tool box is not adequate unless the designer has the expertise of applying the right tool to the protocol of his or her concern. Unfortunately, such a requirement to the protocol designer is often too stringent to be realistic. First, the knowledge required to select a right algorithm or tool is dispersed in the literature and cannot be easily acquired by the designer. Secondly, the designer may be just a novice user of the validation tools and may not be interested in understanding all the available validation algorithms.

Therefore, in addition to the tool box idea, we have proposed another idea called the "intelligent user-interface" to construct a user-friendly protocol validation system. The idea is to develop an acknowledge-based interface that not only managers all the validation algorithms, but also acts as an intelligent assistant to help the protocol designer select and use these algorithms. It is natural to bring in the knowledge-based techniques here because the process of guiding a designer to select and use the most appropriate validation algorithm is basically symbolic.

The structure of such a knowledge-based protocol validation system is illustrated in the Figure 3.

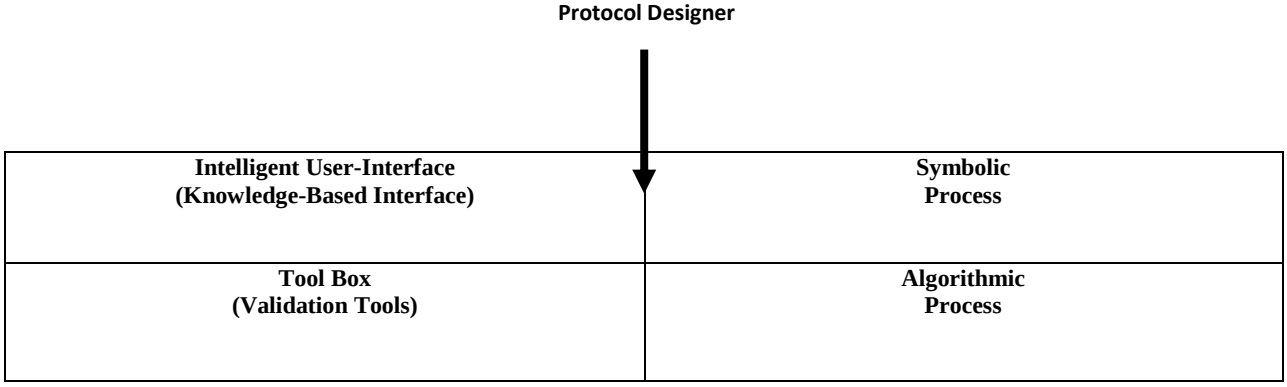

Figure 3: Structure of the knowledge-based protocol validation

Note that the symbolic (non-procedural) process of the system, namely the knowledge-based interface, is one the top of the algorithmic (procedural) processes implemented as a tool box of collection of validation algorithms. This kind of system is now getting attention from the artificial intelligence community that is called the coupled system because both symbolic and algorithmic concerning are coupled in the same system.

In the first stage of our development, we have included the following six validation algorithms in the tool box:

- Fair Progress Validation.
- Maximal Progress Validation.
- Reduced Reachability Analysis.
- Vuong's Reachability Analysis.
- Exhaustive Reachability Analysis.
- Protocol Validation Testing [26].

Among the algorithms listed above, the fifth and sixth are supported by the Probabilistic Transmission Grammar (PTG) validation tool; the first four algorithms are supported by four separated tools. In fact, all these tools are developed by modifying an existing conventional tool called Transmission Grammar. Every tool can accept protocol specification is either TG or PTG. Note that different tools may have different uses and some of them may be quite complicated. Nevertheless, through the guidance and control of the intelligent user-interface, the protocol designer should have no difficulty in utilizing the full power of these tools. Our design of the intelligent user-interface is largely influenced by the idea behind the Conceptual Structures Representation Language (CSRL), a high-level language tuned specifically for implementing diagnostic expert systems [27]. In CSRL, a specific organizational technique called hierarchical classification and a specific problem solving strategy called establish-refine are employed to design a knowledge-based system. We believe that the structure and problem-solving strategy demonstrated by CSRL is quite suited in our domain of building an intelligent user-interface for the protocol validation system. The reasons are argued as follows:

- The decision procedure of which algorithm to use in validating a protocol can be organized as a classification hierarchy of three levels as shown in the following figure.
- The establish-refine control can be used as a search strategy in identifying the protocol under validation with an appropriate algorithm at the tip of the hierarchy (tree).

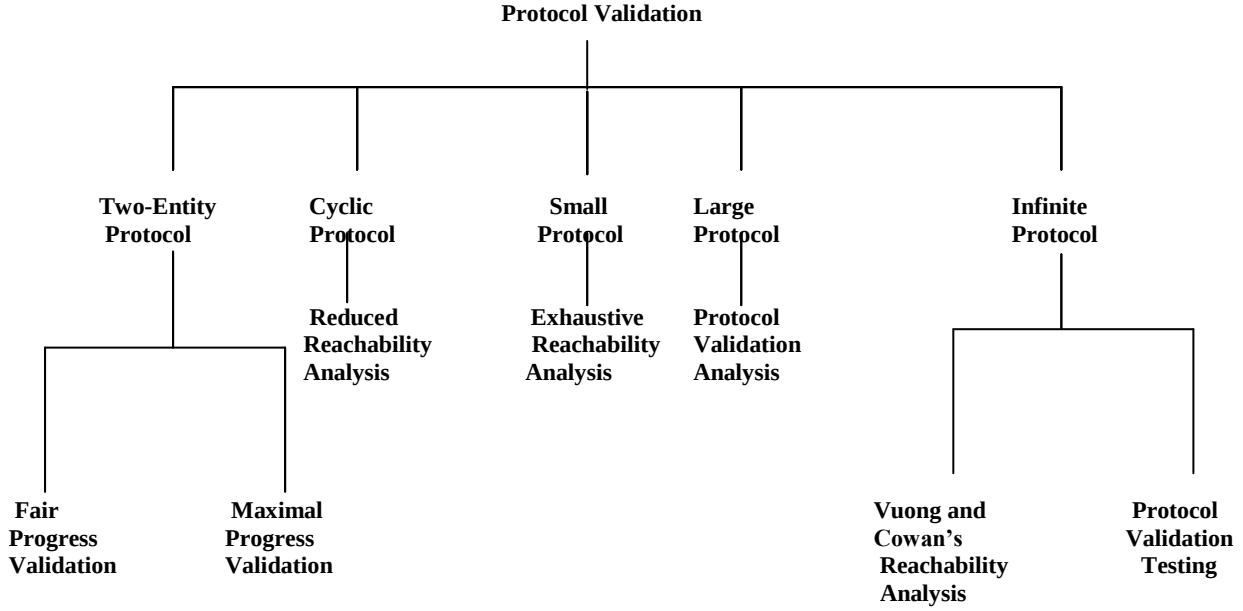

Figure 4: The classification hierarchy (tree) of the intelligent user-interface

To give more details, we briefly describe how this whole process works. From the root of the tree, the specialist (or concept) "protocol validation" first tried to establish itself. If successful, the succeeding refinement of it will pass the control to the second level of specialists. The specialists in the second level then repeat the same process; they first try to establish themselves and if successful, may refine further down the tree after being granted by its super specialist; otherwise, all its sub specialists will be excluded from further consideration. If a specialist at the tip of the hierarchy (the third level of the tree) establishes itself, it essentially means the feasibility of a specific validation algorithm to the protocol. By this process the validation algorithms suited for validating the protocol and their comparative scores can be determined in our domain, the establishment or rejection of a concept is primarily based on (i) the formal protocol specification and (ii) the interaction between the designer and the system during the establish-refine process.

6. CONCLUSION

Protocol validation and verification is a demonstrator of the correctness of a protocol design. A protocol is considered to be correct if it satisfies two kinds of properties, that is**,** syntactic properties (or general properties) and functional properties (or specific properties). The syntactic properties are those desired properties common to all protocols such as freedom from deadlock, completeness and progress. They form the set of implicit requirements that any protocol should fulfill to ensure that its logical structure has no syntactical errors. The absence of syntactical errors, however, does not necessarily imply that the protocol will do what it is supposed to do. In this regard, the functional properties of a protocol define the specific objectives of the protocol. They are usually presented in terms of a set of behaviors, called the communication service, as presented by the protocol user. As mentioned earlier, a protocol can engage in extremely complicated interactions that are beyond human anticipation. A formal analysis is required to ensure that the functional behavior of a protocol conforms to the designer's intention.

To date, while the syntactic properties of protocols have been extensively studied and relatively well understood, much work remains to be done on the functional analysis, also called conform analysis.

The preceding sections have described various aspects of protocol engineering, a rapidly growing area of research in computer communications. A protocol engineering system allows the protocol designer to express the protocol formally, test its specification for correctness (validation and verification), obtain some early indication of how it would perform, compile major parts of the implementation directly from the formal specification and finally, test the resultant implementation to assure that it conforms to the specification (implementation verification or conformance testing). These tasks are performed iteratively until a correct and efficient protocol is developed. The protocol engineering system can also be used by the protocol designer for protocol synthesis and protocol conversion.

As protocol design becomes more and more important due to the proliferation of computer communications, the need to use computer-aided design in the whole life cycle of protocol development becomes obvious. As described above, current protocol design systems do not provide enough support to help the designer make use of a variety of tools available to him or her. We believe that the incorporation of a knowledge-based system can help in those aspects, as they have already done so in other engineering disciplines. A case in point is from the field of software engineering as reported by Mostow and Simon, where knowledge-based systems are used to help automate the whole life cycle of software development. Since protocols are a special class of concurrent programs that are communication-intensive, it is expected that those ideas and techniques developed for software engineering can be applied to protocol engineering as well [28]-[30].

The incorporation of knowledge-based systems into the protocol design process can be done in many ways. For example, program transformation techniques can be used to deriving protocol specifications from given service specifications [31]. Other AI techniques, such as search algorithms and theorem-proving can be used to reduce the global space search and to help correctness proving, respectively, in protocol validation and verification. Therefore, it is expected that both AI techniques and computer-aided software engineering methodologies will play an important role in the future development of protocol engineering.

7. ACKNOWLEDGMENTS

We would like to thank Preston University-Ajman, United Arab Emirates, for providing financial support for this research study. It is a pleasure to thank those at Preston University-Ajman who assisted me: Mr. Raja Sajjad Hussain, Director General, for his guidance, and Professor Mark Langer for his technical assistance.

8. REFERENCES

[1] Denning, D. E. 1987. **"**An Intrusion Detection Model," *IEEE Trans Software Eng* 13(2): 222-232.

[2] Karthikeyan, B., Davor, O. and Carl, A.G. 2002. "Formal verification of standards for distance vector routing protocols,"*JACM*, 49 (4):538-576.

[3] Sunshine, C.A. 1979. "Formal techniques for protocol specification and verification," *IEEE Computer Magazine* 12(9):20-27.

[4] West C.H. 1978. "An automated technique of communications protocol validation," *IEEE Trans Comm.* 26(8):1271–1275.

[5] West, C.H. 1982. **"**Applications and limitations of automated protocol validation," *Proc. 2nd IFIP WG 6.1*:361-373.

[6] Vuong, S. T. and Cowan, D. D. 1982. "Reachability Analysis of Protocols with non-FIFO Channels," *Proc. COMPCON*: 49-57.

[7] Choi, T.Y., and Miller, R.E. 1983. "A decomposition method for the analysis and design of finite state protocols*," Proc. ACM/IEEE Data Comm*: 167-176.

[8] Chow, C.H. 1985. "A discipline for the verification and modular construction of communication Protocols," Ph.D. dissertation, University of Texas, Austin Texas.

[9] Lam, S.S., and Shankar, A.U. 1984. "Protocol Verification via projections," *IEEE Trans Software Eng* 10(4):325-342.

[10] Blumer, T.P., and Sidhu, D.P. 1986. "Mechanical verification and automatic implementation of communication protocols," *IEEE Trans Software Eng.*12 (8):827-842.

[11] Ansart, J.P. 1985. **"**Issues and tools for protocol specification. In Distributed Systems, Methods and Tools for Specification," *Springer Verlag, Berlin Germany*: 481-538.

[12] Rudin, H., and West, C.H. 1982. "An improved protocol validation technique," *Computer Networks* 6(2):65-73.

[13] Gouda, M.G., and Han, J.Y. 1985. "Protocol validation by fair progress state exploration," *Computer Networks & ISDN Systems* 9:353-361.

[14] Zhao, J.R., and Bochmann, G. V. 1986. **"**Reduced reachability analysis of communication protocols: a new approach," *6th Int'l Workshop on PSTV*: 234-254.

[15**]** Gouda, M.G. and Yu, Y.T. 1984. "Protocol validation by maximal progress state exploration," *IEEE Trans Comm.* 32(1):94-97.

[16] Gouda, M.G., and Yu, Y.T. 1984. "Synthesis of Communicating Finite-State Machines with guaranteed progress," *IEEE Trans Comm.* 32(7):779-788.

[17] Itoh, M., and Ichikawa, H. 1983. "Protocol verification algorithm using reduced reachability analysis," *Trans. of IEICE Tran* E66 (2):88-93.

[18] Brand, D., and Zafiropulo, P**.** 1983. "On Communicating finite state machines," *J ACM* 30(2):323-342.

[19] Kakuda, Y., Wakahara, Y., and Norigoe, M. 1986. "A new algorithm for fast protocol validation," *Proc. IEEE COMPSAC*: 228-236.

[20] Holzmann, G.J. 1985. "Tracing Protocols," *AT&T Tech. Jour.* 64(10):2413-2434.

[21] Holzmann, G.J. 1987. "Automatic protocol validation in Argos, assertion proving and scatter searching," IEEE *Trans Software Eng* 13(6):683-696.

[22] West, C.H. 1986. **"**Protocol Validation by Random State Exploration," *Proc. 6th IFIP WG* 6.1:233-242.

[23] Vuong, S.T., Hui D.D., and Cowan, D.D. 1986. "Valira - A Tool for Protocol Validation via Reachability Analysis," Proc. *6th Workshop on PSTV:* 35-42.

[24] Lu, **C.S.** 1986. **"**Automated validation of communication protocols," *Ph.D. dissertation, Ohio State University, Columbus, Ohio.*

[25] Purushothaman, S., and Subrahmanyam, P.A. 1987. "Reasoning about probabilistic behavior in concurrent systems," *IEEE Trans Software Eng* 13(6):740-745.

[26] Lin, F.J., Chu, P.M., and Liu, M.T. 1987. "Protocol verification using reachability analysis: the state explosion problem and relief strategies," *Proc. ACM SIGCOMM* 17(5):126-135.

[27] Bylander. T, Mittal.S. 1986. "CRSL: A Language for Classificatory Problem Solving and Uncertainty Handling," *AI Magazine* 7(3):66-77.

[28] Mostow, J. 1985. **"**Toward Better Models of the Design Process," *AI Magazine* 6(1):44-57.

[29] Mostow, J. 1985. **"**Foreword What is AI? And what Does It Have to Do with Software Engineering?" *IEEE Trans Software Eng* 11(11): 1253-1256.

[30] Simon, H.A. 1986. "Whether Software Engineering Needs to be Artificially Intelligent," *IEEE Trans Software Eng* 12 (7): 726-732.

[31] Balzer, R. 1985. "A 15 Year Perspective on Automatic Programming," *IEEE Trans Software Eng* 11(11):1257-1268.