ALGEBRA OF PROTOCOL STRUCTURES BASED ON THE THEORY OF PETRI NETS

N. A. Anisimov

Avtomatika i Vichislitel'naya Tekhnika,
Vol. 21, No. 1, pp. 9-15, 1987

UDC 681.324

Methods of formalization of protocol control structures are described. An algebra of protocol structures is suggested on the basis of the algebra of regular Petri nets supplemented with the operation of destruction. A graphic structural representation of protocol structures is introduced. The formalization method is illustrated by an application to recommendation X.25/3 protocol. A method reducing the protocol verification to a verification of protocol components is proposed. The concepts of hierarchical relationship of protocols are defined in formal terms, and a rule of hierarchical composition is introduced.

INTRODUCTION

The efficiency of software for information and computer networks (ICN) [1] largely depends on the existence of formal methods supporting the software development. The distributed nature and complex hierarchical structure of ICN software and the trend for increasing complexity of the services and protocols require creating the fundamental theory and developing its formal tools. These tools should, first, describe adequately the structural and behavioral aspects of the protocols and, second, be based on a developed theory supporting the formalization, verification, synthesis, and implementation of protocols.

Vast experience has been accumulated in protocol formalization and verification [2]. One popular approach is to represent complex protocols as sets of simpler protocols. Protocol composition methods are known, which use generalized automata [3], aggregates [4], and networks of logical asynchronous processes [5]. Conversely, protocol simplification procedures are used for validation, including decomposition methods [6] and protocol projections [7].

This paper proposes using as a formal basis of protocol theory Petri nets [8] whose advantages are well known in the description of asynchronous and parallel processes. Petri nets are equipped with an extensive inventory of operation tools, including methods of analysis [8], results of computer simulations, and network protocols [10].

A formal description of the entire set of protocols is an important component in developing ICN software. It is a feature of most existing protocols and recommendations that control aspects are separated from operations on parameters and variables. This separation advances formalization of protocol structures and multilevel protocol systems as a separate problem.

The complexity of real protocols is due to their complex structure. It is customary to describe protocols informally in parts, dividing them into phases, classes, subprotocols, etc. This approach seems promising also in formal protocol descriptions. Methods of description of protocol parts and formal rules for integrating parts into the final protocol are necessary. With this approach the rigorous description can be combined with clarity and comprehensibility.

1. ALGEBRA OF PROTOCOL STRUCTURES

We proceed from an approach [8] where the basic concepts and definitions are given in terms of Petri net theory (PNT) and form a protocol interpretation of PNT. This provides a formal basis for protocol theory and makes it possible eventually to transfer some of the PNT results into it.

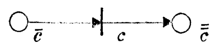

Fig. 1. Elementary protocol structure.

We will introduce an algebra of protocol structures built on the basis of an algebra of regular nets [10], expanded by inclusion of operations of destruction. The definitions of protocol structure algebra will be given as interpretations of the elements and operations of the algebra of regular nets. For brevity, we will speak of protocols rather than protocol structures, except for cases where this might cause confusion.

Assume that a finite set of names of protocol commands $C=\{c_1,c_2,\dots,c_n\}$ is fixed. Each command name has a prefix which can be one of the symbols $\downarrow$ or $\uparrow$ designating the sending or receiving of a command, respectively.

An elementary protocol is defined as a net of the form shown in Fig. 1. Here, $c \in C$ and $\bar{c}$ and $\bar{\bar{c}}$ are the head and tail places of the net corresponding to the initial and terminal stage of the elementary protocol, respectively. Operations on protocols are defined as interpretations of similar operations on regular nets, which are well known [10] and therefore are described informally. Each protocol is associated with a regular net whose head and tail places are interpreted as initial and terminal states of the protocol. The protocol class generated by operations on protocols and the class of elementary protocols will be called the class of regular protocols. If the operation of superposition is not used, such a class will be called the class of primitive protocols.

Let N, N_1 and N_2 be regular nets and P, P_1 and P_2 their corresponding protocols.

Operation of iteration "*". The operation of iteration on a protocol P_1 constructs a protocol $P = *(P_1)$ such that $N = *(N_1)$. The protocol P is cyclic and its initial state coincides with the terminal state. The operation of iteration can be used to describe cyclic procedures which occur in all protocols.

The operation of attachment ";". The operation of attachment of protocols P_1 and P_2 combines them into a protocol $P = (P_1;P_2)$ such that $N = (N_1;N_2)$. The initial state of the protocol P is the initial state of the protocol P_1, and its terminal state is the terminal state of the protocol P_2. The operation of attachment formalizes the rule of composition of two protocols, where one can be initiated only after the other is completed. For example, data transmission in a protocol can start only after the establishment of communication is completed.

Operation of exclusion "□". The operation of exclusion of protocols P_1 and P_2 combines them into a protocol $P=(P_1 \square P_2)$ such that $N=(N_1 \square N_2)$. The initial and terminal states of the protocols are merged. The operation of exclusion constructs a configuration with alternative execution of procedures. This operation describes the fundamental conflict situation which occurs in many protocols.

Operation of superposition ",". The operation of superposition of protocols P_1 and P_2 combines them into a protocol $P = (P_1,P_2)$ such that $N = (N_1,N_2)$. The sets of the elements of the nets representing the protocols are combined. This operation can be used to describe parallelism and independence of execution of protocol procedures. For example, the procedures for transmission of expedited and ordinary data blocks in many protocols are independent and can be represented as parallel processes.

Before defining the next operation we must generalize the definition of the superposition to extend it to the case where the nets being superimposed have multiple transition symbols. Assume that the net N_1 has k transitions with symbol a and the net N_2 has l such transitions. First, the system numbers in each net, and marks with a subscript. The transitions with the symbols a: $\{a_1,a_2,\dots,a_k\}$ and $\{a_1,a_2,\dots,a_l\}$. Then, each transition a_i in the net N_1 is copied l times, and each copy is marked as a_i^j, where $j \in 1,l$.

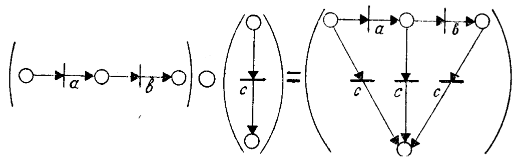

Fig. 2. Operation of primitive net destruction.

Likewise, a_i are marked in the net N_2. After this, the subscripts and superscripts for each transition a_i^j of the net N_2 are transposed. After similar sequences of actions are performed on other multiple transition symbols and the nets have no multiple transitions, the operation of superposition of the resulting nets is carried out.

The operation of destruction "O". The operation of destruction has no analog in the algebra of regular nets. It is introduced to describe the commonprocedures, such as clearance, restart, disconnect, etc., when a protocol in any state can carry out the required procedure. A similar operation has been used in [6] for protocol decomposition.

The operation of destruction of regular nets is defined as follows. Assume that N_2 is a net with a single head place and the net N_1 is a primitive net, i.e., one constructed without superposition. The net $N=(N_1 O N_2)$ is constructed in two steps. First, the head place of the net N_2 and each of its incident transitions with arcs are copied as many times as there are places in the net N_1. Then, each place of the net N_1 is merged with one of the copies of the head place of the net N_2. The head place of the net $N = (N_1 O N_2)$ is now identified as the head place of the net N_1 and the tail place as the tail place of N_2. The net N_1 will be called the destructed net and the net N_2 the destructing net. Figure 2 illustrates the operation of destruction.

The operation of destruction can be defined for a broader class of destructed nets, namely, when the destructed net is represented in the form $N_1 = (A, B)$ and the regular nets A and B have no common places. First, the nets $(A O N_2)$ and $(B O N_2)$ are constructed, which are then combined by means of superposition. Figure 3 shows an example of destruction of nets of this kind.

The operation of destruction of protocols P_1 and P_2 combines them into a protocol $P=(P_1 O P_2)$ such that $N=(N_1 O N_2)$. The initial state of the protocol P is the initial state of the protocol P_1, while its terminal state is the terminal state of P_2.

This algebra, as well as the algebra of regular Petri nets, is susceptible to a formula representation allowing a compact expression of complex protocol structures. The descriptions in this algebra not only determine the external behavior of a protocol entity but can also be used to define its internal structure, providing at the same time a basis for implementation. This latter quality and the possibility of describing the destruction procedures distinguish this algebra from other algebraic approaches.

2. GRAPHIC REPRESENTATION OF PROTOCOL STRUCTURES

Despite the compactness of the formula descriptions of structure, graphic representations are cumbersome. The complexity of the resulting Petri netis due to the large number of transitions generated by the operation of destruction. In order to eliminate this shortcoming, a structural representation of Petri nets of this class called macronets is suggested.

The entire set of the place of the macronet is divided into two nonintersecting nets of simple and composite places. Simple places are similar to places of Petri nets. A composite place is comprised of a separate macronet, admitting an arbitrary degree of nesting. Assume that a net $N=(N_1 O N_2)$ is given. It is represented graphically as a net N_2 with a composite head place which contains the net N_1. By consecutive application of this rule to each operation of destruction in the formula one obtains a macronet which is a regular net with the operation of destruction.

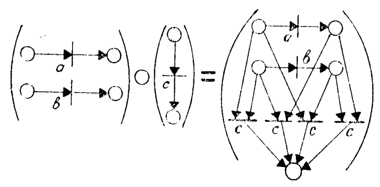

Fig. 3. Operation of parallel net destruction.

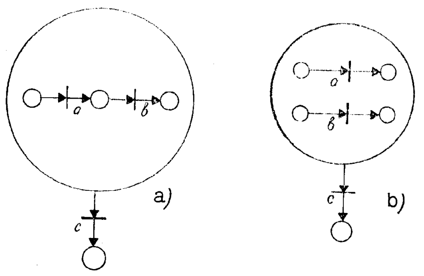

Fig. 4. Examples of macronets.

The rules of functioning of macronets are defined as follows. The condition for activating a transition which has a composite input place is the presence of at least one label in the internal net. If the transition which has a composite input place is triggered, a label is extracted from the corresponding net wherever it is located. If the internal net contains parallel fragments, a label is extracted from each fragment. If the transition has a composite output place, when it is activated the label is placed in the head place (or places) of the internal net.

Figures 4a and 4b give the graphic expressions of the nets represented in Figs. 2 and 3, respectively.

EXAMPLE

For an illustration of the use of the algebra of protocol structures we will describe the structure of the batch layer of Recommendation X.25 for virtual channels at the side of the terminal data equipment. Assume $C=C_1 \cup C_2 \cup C_3 \cup C_4 \cup C_5 \cup C_6$, where $C_1=\{\downarrow CR, \uparrow CR, \downarrow CA, \uparrow CA\}$, $C_2=\{\downarrow CLR, \uparrow CLR, \downarrow CLC, \uparrow CLC\}$, $C_3=\{\downarrow DT, \uparrow DT, \downarrow RR, \uparrow RR, \downarrow RNR, \uparrow RNR\}$, $C_4=\{\downarrow I, \uparrow I, \downarrow IC, \uparrow IC\}$, $C_5=\{\downarrow RS, \uparrow RS, \downarrow RSA, \uparrow RSA\}$, $C_6=\{\downarrow RST, \uparrow RST, \downarrow RSTC, \uparrow RSTC\}$ are the sets of names of commands of the procedures of establishing connection, interrupting connection, controlling the flow, transmitting interruption, clearance, and restart, respectively. The structure of the procedures is defined by the following formulas:

1. Establish connection:

$$N_1=((\downarrow CR;\ (\uparrow CA \square (\uparrow CR;\ \uparrow CA))) \square (\uparrow CR;\ \downarrow CA))$$

2. Disconnet:

$$N_2=((\downarrow CLR;\ (\uparrow CLC \square \uparrow CLR)) \square (\uparrow CLR;\ \downarrow CLC))$$

3. Flow control:

$$N_3=(*(\downarrow DT \square \uparrow DT \square \downarrow RR \square \uparrow RR);\ *(\downarrow RNR;\ \downarrow RR);\ *(\uparrow RNR;\ \uparrow RR))$$

4. Interrupt transmission:

$$N_4=(*(\downarrow I;\ \uparrow IC),\ *(\uparrow I;\ \downarrow IC))$$

5. Clearance:

$$N_5=((\downarrow RS;\ (\uparrow RS \square \uparrow RSA)) \square (\uparrow RS;\ \downarrow RSA))$$

6. Restart:

$$N_6=((\downarrow RST;\ (\uparrow RSTC \square \uparrow RST)) \square (\uparrow RST;\ \downarrow RSTC))$$

The structure of the entire procedure is described by the formula

$$N=*(*((N_1;*((N_3, N_4) O N_5)) O N_2) O N_6)$$

Figure 5 gives the graphic representation of the structure. In [11] the structure of the transport protocol was also described.

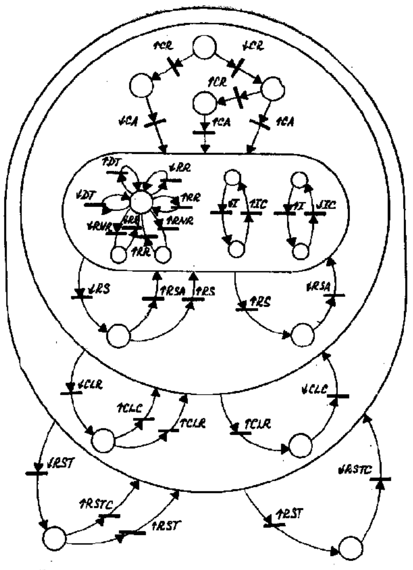
Fig. 5. Structure of the protocol of Recommendation X.25/3.

4. VALIDITY OF PROTOCOL STRUCTURES

Properties of protocols depend on the properties of their structure. It is important, therefore, to test protocol structures for validity. With an algebraic approach the validity analysis of complex structures can be reduced to analysis of their components.

Protocol verification should be based on a model consisting of protocol entities and the environment. A protocol structure defined by a regular net determines the behavior of a protocol entity. In the subsequent discussion we proceed from a model and results of the theory of protocol projections [7]. In terms of these theories, the destruction is a well-formed operation, because, according to the definition, the state of destruction of a protocol entity is immediately attainable from any state of the destructed protocol. As follows from the main result of the theory of projections, the property of being well-formed guarantees that the properties of being live and safe will coincide for the entire protocol and its components. The analysis of these properties for a protocol defined by the operation of destruction is thus reduced to the analysis of the properties of the destructing and destructed protocols.

For a regular protocol the notion of projection with respect to the degree of nesting is defined. The projection of the first degree of the protocol P is defined as the initial net in which all composite places are replaced by simple places. A projection of the degree k, where $k > 1$, is defined as the net which is the internal net for the projection of a degree in which the proper composite places are replaced by simple places. By virtue of the property of being well formed, the analysis of a protocol for the properties of being live and safe is reduced to the analysis of these properties in all its projections.

The higher the degree of structuring, the greater the number of simple components into which a protocol is divided and the easier their analysis.

5. HIERARCHICAL COMPOSITION OF PROTOCOLS

The software of an ICN has a hierarchical structure and is divided into layers, each operating according to its own protocol. The set of protocols organized in this fashion forms a hierarchical protocol system. Within a layer it is sometimes convenient to separate sublayers and, accordingly, divide a protocol into subprotocols. This makes it necessary to define formally the notions of hierarchical dependency of protocols and formulate the rules of hierarchical composition of protocol structures.

Informally, the hierarchical dependency of protocols is seen in the fact that the

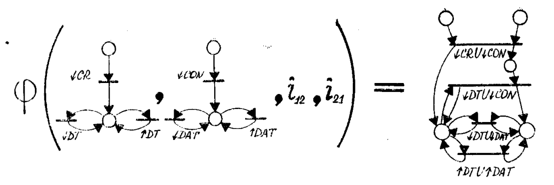

Fig. 6. Operation of hierarchical composition.

execution of the command of a protocol of a certain layer corresponds to the execution of one or more commands of another layer. Suppose that two protocols P_1 and P_2 are given which are defined on the set of names of commands C_1 and C_2. The hierarchical relationship of the protocols P_1 and P_2 is defined by the mappings $i_{12}: C_1 \to F(C_2)$ and $i_{21}: C_2 \to F(C_1)$, where F(C) denotes the set of primitive formulas of protocol algebra on the set of names C. Indeed, a command of a protocol is associated with a certain structure constructed from commands of another protocol, which determines the sequence of their execution.

The operation of hierarchical composition of the protocols P_1 and P_2 with respect to the mappings i_{12} and i_{21} combines them into the protocol $P = \varphi(P_1, P_2, i_{12}, i_{21})$, which is built according to the following rules. First, in the formula of the protocol P_1 each occurrence of the symbol $c_1 \in C_1$ is replaced by the formula $i_{12}(c_1)$, in which every name $c_2 \in C_2$ is substituted by the name $c_1 \cup c_2$. This involves a simple generalization which allows operating with sets of commands on a par with their names. The formula of the protocol P_2 is transformed in a similar way. Finally, the nets corresponding to the resulting formulas are combined by means of the operation of superposition.

Figure 6 gives an illustration of the execution of hierarchical composition of simple protocols $P_1 = (\downarrow CR; *(\downarrow DT \square \uparrow DT))$ and $P_2 = (\downarrow CON; *(\downarrow DAT \square \uparrow DAT))$ relative to the mappings i_{12} and i_{21}:

$$\begin{aligned} i_{12}:\ & \downarrow CR \to \downarrow CON & \qquad i_{21}:\ & \downarrow CON \to (\downarrow CR; \downarrow DT) \\ & \downarrow DT \to (\downarrow CON \square \downarrow DAT) & & \downarrow DAT \to \downarrow DT \\ & \uparrow DT \to \uparrow DAT & & \uparrow DAT \to \uparrow DT \end{aligned}$$

The transmission and reception of all the commands of the protocol P_2 are carried out by means of the commands ↓DT and ↑DT of the protocol P_1. In addition, the execution of the command ↓CON of the protocol P_2 should correspond to the consecutive execution of the commands ↓CR and ↓DT of the protocol P_1.

The hierarchical composition of protocols produces a situation where the activation of a transition can correspond to the concurrent execution of two commands of different layers. This dispenses with the need for developing and implementing a special interface and allows describing the multilayer logical structure of ICN software as a single net and implementing it as one software (or hardware) module.

CONCLUSIONS

The paper proposes an algebra which produces a strict, compact, and transparent description of the structures of complex protocols. An advantage of this algebraic approach is that it brings together the processes of development, formalization, and comprehension of protocols, i.e., the process of understanding a formal description of the protocol replicates the process of protocol development and formalization.

For protocol verification this method reduces the validation of complex protocols to the testing of their constituents. The similarity of the structures of different protocols appears to make it possible to identify a set of standard "valid" structures and to use them as building blocks to construct protocols with guaranteed good qualities.

REFERENCES

1. E. A. Yakubaitis, Information and Computer Networks [in Russian], Financy i Statistika, Moscow, 1984.

2. S. S. Zaitsev and N. A. Muradyan, Methods of Validation of the Functioning of Network Protocols [in Russian], Preprint of the Scientific Council on Cybernetics, USSR Academy of Sciences, Moscow, 1983.

3. V. G. Lazarev, "Elements of the finite state theory of protocols of computer networks," in: Proceedings of the Fourth National Seminar on Control Problems for Communication Networks and Nodes [in Russian], pp. 68-71, Nauka, Moscow, 1984.

4. G. I. Pranyavichis, Models and Methods of Analysis of Computer Systems [in Russian], Mokslas, Vilnius, 1984.

5. S. V. Rotanov, "An approach to analysis of protocol validity," AVT [Automatic Control and Computer Sciences], no. 4, pp. 17-19, 1982.

6. S. T. Vuong and D. D. Cowan, "A decomposition method for the validation of structured protocols," INFOCOM 82 Proc. Conf., pp. 209-220.

7. S. S. Lam and A. U. Shankar, "Protocol verification via projections," IEEE Trans., Software Eng., vol. SE-10, no. 4, pp. 325-342, 1984.

8. J. Peterson, Theory of Petri Nets and System Simulation [Russian translation], Mir, Moscow, 1984.

9. N. A. Anisimov, Methods of Formal Description, Verification, and Implementation of Network Protocols with the Aid of Petri Net Theory [in Russian], Preprint of the Institute of Automatics and Control Problems, Far East Scientific Center, USSR Academy of Sciences, Vladivostok, 1984.

10. V. E. Kotov, "Algebra of regular Petri nets," Kibernetika, no. 5, pp. 10-18, 1980.

11. N. A. Anisimov, "Algebraic approach to formalization and verification of protocol structures on the basis of the theory of Petri nets," in: Packet Switching Computer Networks: Proceedings of the Fourth National Conference, Part 1 [in Russian], pp. 158-162, Riga, 1985.

12 December 1985