RECURSIVE PROTOCOL DEFINITION ON THE BASIS OF THE THEORY OF PETRI NETS

N. A. Anisimov

Avtomatika i Vychislitel'naya Tekhnika,
Vol. 21, No. 5, pp. 3-7, 1987

UDC 681.324

The author introduces the algebra of regular macronets (RMN), which is an extension of the algebra of regular Petri nets and is intended for describing protocol structures. To describe protocols that employ similar procedures for executing dissimilar functions, the author proposes a formal model that is called a recursive RMN and that constitutes a compact notation of an RMN of a certain class. The use of the model is illustrated by the example of the X.25/3 protocol, where the disconnect, reset, and restart procedures are defined by means of a single structure.

Protocols are of fundamental importance in computer and data networks [1]. As a rule, actual protocols have a complex structure and are defined by means of a set of procedures that are intended for executing certain functions. At the same time, protocols frequently employ similar procedures to implement dissimilar functions. For example, data-transmission procedures employing the window mechanism at the transport, network, and link layers have a high degree of similarity. The disconnect, reset, and restart procedures in Recommendation X.25/3 are similar in many respects. These and other facts indicate that it is possible to define a set of standard abstract protocols which could be used to execute various functions in the process of constructing and describing specific protocols. This paper will attempt to formally substantiate an approach of this type, utilizing recursive definition of protocols within the confines of a given layer. For this purpose, we propose a formal model, developed within the framework of the theory of Petri nets, that is a further elaboration of the apparatus of regular macronets [2]. The author has utilized the idea of recursive definition of Petri nets [3,4].

DEFINITION OF MACRONETS

To represent protocol structures, paper [2] informally introduced macronets, comprising regular Petri nets with the operation of "disruption." Below we will give a more rigorous definition, based on the definition of regular Petri nets [5].

Definition 1. A regular macronet (RMN) is a collection $MN=\langle R, W, N_0, \rho\rangle$, where $R=\{N_0, N_1,\ldots,N_n\}$ is a finite set of regular nets $N_i=\langle P_i, T_i, I_i, O_i, M_{0i}\rangle$, that do not have common elements, i.e., $T_i\cap T_j=\varnothing$ and $P_i\cap P_j=\varnothing$ for $i\neq j$; $W=\{p_1, p_2,\ldots, p_n\}$ is a finite set of nonterminal symbol-positions such that for $\forall i\exists j$: $p_i\in P_j$; $N_0\in R$ is an external regular net; ρ: $W\to R\setminus\{N_0\}$ is a one-to-one mapping that specifies a definition of nonterminal symbol-positions.

In graphic terms, an RMN is a regular net that has composite positions, inside of which there may be independent macronets [2]. Composite positions correspond to nonterminal symbol-positions. It is assumed that recursion is not allowed in the definition of composite positions.

The head position h(MN) of an RMN is what we call the head position $h(N_0)$ of regular net N_0 if it is simple. If it is composite, then we consider the head position of the internal net $\rho(h(N_0))$. The process continues until a simple head position is discovered, and this is designated as the head position of macronet MN.

In defining operations over RMN, we will confine ourselves for simplicity to the case in which all nets N_1 are primitive, i.e., they are generated without the operation

of superposition. The iteration, adjunction, and exclusion operations "*", ";", and "□" of RMN involve the execution of the corresponding operations with external nets N_0.

The marking operation n(MN) adds n markers to the head position of the external net N_0. If this position is composite, then n markers are also added to the head position of the internal net. The process is continued until markers have been added to a simple head position.

Execution of the "disruption" operation $MN=(MN_1 \circ MN_2)$ involves declaring head position $h(MN_2)$ to be a composite position, and placement at it of net MN_1 to be "disrupted," i.e.,

$$(MN_1 \circ MN_2)=\langle R_1 \cup R_2,\ W_1 \cup W_2 \cup \{h(MN_2)\},\ N_{02},\ \rho_1 \cup \rho_2 \cup \{(h(MN_2), N_{01})\}\rangle.$$

The operational rules for RMN are defined by means of two fundamental operations over nets N_1, namely placement of a marker at the head position, and removal of a marker from the net, regardless of the position in which it is located. We denote the input and output sets of transition positions t as ${}^\circ t=\{p/I(p,t)=1\}$ and $t^\circ=\{p/O(p,t)=1\}$. Activation of transition $t \in T_i$ is supplemented by the following rules. If the input transition positions t include composite ones, i.e., $p \in {}^\circ t \cap W$, then a marker is extracted from the internal net $\rho(p)$. If the output positions include composite ones, i.e., $p \in t^\circ \cap W$, then a marker is inserted in the corresponding internal net $\rho(p)$.

Graphically speaking, an RMN can also be represented by a set of nets R, each composite position of which has an indicator or pointer to an internal net from R. In the protocol interpretation, each net N_1 corresponds to a certain procedure, which can be initiated or disrupted by other procedures.

Externally, RMN have many features in common with hierarchical nets [5], macrographs [6], and complex hierarchically structured nets [7], but they differ significantly from them in terms of their operating rules.

RECURSIVE MACRONETS

The similarity of some procedures of protocols whose structure is specified by a macronet, expresses itself in the fact that the corresponding nets have the same topology. The set of these nets can be specified by a single net in which certain values are associated with markers, as is the case, e.g., in colored Petri nets [3].

Assume that X is a variable that is specified on the set of values Dom X. With each net N_i in Definition 1 we associate variable X_i, and we denote the resultant net as $N'_i=\langle N_i, X_i\rangle$. This interpretation specifies a set of nets $N^*_i=\{\langle N_i, a\rangle / a \in Dom\, X_i\}$. We denote a net from this set by $N_i(a)$. Each position $p_i \in P_j$ and transition $t_i \in T_j$ will correspond to sets p^*_i and t^*_i, whose elements are denoted by $p_i(a)$ and $t_i(a)$ respectively. Let $W^* = \bigcup_{p \in W} p^*$, $R^*=\bigcup_{N_i \in R} N^*_i$, $\rho'\colon W^* \to R^*$.

Definition 2. A recursive regular macronet is a collection $RMN=\langle R', W, N_0(a), \rho'\rangle$, where $R'=\{N'_0, N'_1, \ldots, N'_n\}$ and $N_0(a) \in R^*$.

Thus, a recursive RMN is a set of regular nets, in which the markers are identified by values from a certain particular set. It should be pointed out that RMN are a particular case of recursive RMN for $|Dom\, X_i|=1$ for $\forall i \in 0, n$. Insertion of a marker at the head position of net $N_i(a)$ is equivalent to insertion of a marker with value a at the head position of net N_i. The same is true of removal of a marker. The operating rules for recursive RMN are modified as follows. Assume that, in net $N'_i=\langle N_i, X_i\rangle$, transition $t(a)$ with $a \in Dom\, X_i$ is activated, which carries markers with value a from input to output positions. Then, if the input positions of transition t include a composite one (i.e., $p \in {}^\circ t \cap W$), then a marker is removed from net $\rho'(p(a))$. If the output positions of transition t include a composite one ($p \in t^\circ \cap W$), then a marker is placed at the head position of net $\rho'(p(a))$.

A recursive RMN can be represented graphically by a set of nets N'_i and by function ρ'. At the same time, to keep the protocols clear and tractable, it can be expanded into an RMN and represented by a single net.

PROTOCOL INTERPRETATION OF RECURSIVE RMN

Our protocol interpretation of recursive RMN will be illustrated using the structure of Recommendation X.25/3 for the virtual-circuit mode for data terminal equipment, whose formal description on the basis of RMN was provided in [2]. We introduce the following notation. Assume that x identifies some protocol command. Using this symbol, we can define the following set of command names: $\{\downarrow x\_R, \uparrow x\_R, \downarrow x\_C, \uparrow x\_C\}$. Here the prefix denotes transmission ($\downarrow$) or reception ($\uparrow$), while the suffix denotes request (R) or confirmation (C).

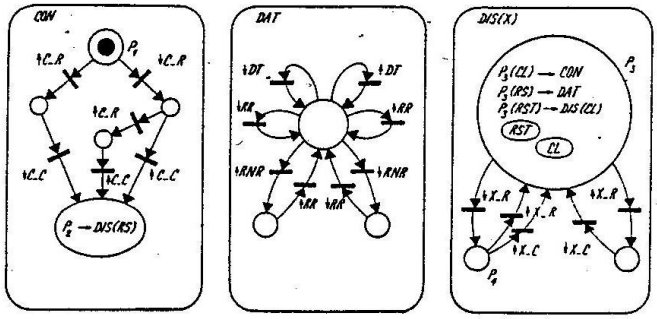
Assume that we have fixed the set of command names $C=C_1 \cup C_2 \cup C_3 \cup C_4 \cup C_5$, where $C_1=\{\downarrow C\_R, \uparrow C\_R, \downarrow C\_C, \uparrow C\_C\}$ is the set of command names for the connection-establishment procedure; $C_2=\{\downarrow CL\_R, \uparrow CL\_R, \downarrow CL\_C, \uparrow CL\_C\}$ is for the disconnection procedure; $C_3=\{\downarrow DT, \uparrow DT, \downarrow RR, \uparrow RR, \downarrow RNR, \uparrow RNR\}$ is for the data-transmission procedure; $C_4=\{\downarrow RS\_R, \uparrow RS\_R, \downarrow RS\_C, \uparrow RS\_C\}$ is for the reset procedure; and $C_5=\{\downarrow RST\_R, \uparrow RST\_R, \downarrow RST\_C, \uparrow RST\_C\}$ is for the restart procedure. For purposes of simplicity, we will not consider the interrupt-transmission procedure.

We define the abstract disruption procedure as $DIS(x)=*((\downarrow x\_R; (\uparrow x\_C \square \uparrow x\_R)) \square (\uparrow x\_R; \downarrow x\_C))$, where x functions as a formal variable defined on the set $Dom\, x=\{CL, RS, RST\}$. Then the structure of the protocol can be represented as the net $N=1(((CON; (DAT \circ DIS(RS))) \circ DIS(CL)) \circ DIS(RST))$, where $CON=((\downarrow C\_R; (\uparrow C\_C \square (\uparrow C\_R; \uparrow C\_C))) \square (\uparrow C\_R; \downarrow C\_C))$ is the structure of the connection-establishment procedure, while the structure of the data-transmission procedure has the form $DAT=(*(\downarrow DT \square \uparrow DT \square \downarrow RR \square \uparrow RR); *(\downarrow RNR; \downarrow RR); *(\uparrow RNR; \uparrow RR))$. Here, the single standard structure DIS is used to specify the structure of three different procedures (disconnection, reset, and restart).

The accompanying figure depicts the recursive RMN describing the structure of the protocol. The RMN consists of three regular nets. The variable x is associated with the net DIS, and markers in it can assume values from the set $Dom\, x=\{CL, RS, RST\}$. Positions p_2 and p_3 are composite. Each position has an associated mapping, which defines an internal net, in accordance with the value of the marker. The initial marking of position p_3 has two markers: CL and RST. The net can operate as follows. Upon activation, e.g., of transition $\downarrow x\_R$ with x = CL, the marker with identifier CL is moved from position p_3 to p_4. Since p_3 is a composite position, a marker is removed from the internal net (from position p_1), which is provided by the net CON in this case. Further, upon activation of transition $\uparrow x\_C$ with x = CL, marker CL is moved from position p_4 to p_3. The marker is placed at the head position of the internal net CON. This sequence of activation of transitions is interpreted as execution of the disconnection procedure. Execution of this procedure disrupts the execution of the connection-establishment procedure, corresponding to removal of a marker from the net. If, for example, the connection were in the data-transmission phase at the instant of disconnection (with a marker at position p_2 of CON, and hence with a marker in net DIS(RS)), then the reset and data-transmission procedures would also be disrupted, since p_2 is a composite position.

The above example demonstrates that our model is suitable for representing protocols at a fairly high level of abstraction, i.e., on the control-structure level. At the same time, the model can also be used for complete definition of a protocol. Indeed, the complete description of a protocol can be specified by an interpreted Petri net, in which positions are associated with attribute vectors, while transitions are associated with operations over them [9]. In the protocol interpretation, attributes are treated as variables and parameters of a protocol, as fields of protocol blocks. Returning to our example, and comparing the complete disconnection, reset, and restart procedures, we can observe that they have the same formats of protocol blocks and timing mechanisms (i.e., timer switch-on, switch-off, and completion procedures). At the same time, these procedures differ in terms of the parameter values - the time-outs and the values of the fields of protocol blocks. These differences can be taken into account by introducing additional variables into the abstract procedure. For example, in the disruption procedure DIS, in addition to the variable x we create a variable that defines the value of the time-out, and possibly other variables as well.

Although it was shown above that our proposed approach can be used in general to formalize existing protocols, it is more effective to employ it in developing and describing new protocols. A protocol developer who has available a set of abstracts, frequently used, well-verified (and, perhaps, implemented) procedures can use them to construct complex realistic protocols. He can employ the same procedure several times to



Structure of X.25/3 protocol.

implement different protocol functions. The idea of this approach is very similar to that of assembly programming [10], which ensures the soundness of the development process. Even at present, some results obtained using this approach are available [11].

CONCLUSIONS

We have proposed a formal model that makes it possible to simplify the description and comprehension of protocols of a certain class. Once the developer has understood the structure and functioning of an abstract procedure, he can readily comprehend the operation of the corresponding specific procedures. This makes it possible to simplify, and evidently to reduce, the work involved in subsequently implementing the protocol. No less important is the fact that the proposed approach simplifies the task of protocol verification, since the fact that the abstract protocol procedure is correct implies that the corresponding step of specific protocols is also correct [2], and protocol procedures can be verified and investigated independently of one another.

REFERENCES

1. E. A. Yakubaitis, Computer and Data Networks [in Russian], Finansy i statistika, Moscow, 1984.
2. N. A. Anisimov, "Algebra of protocol structures based on the theory of Petri nets," AVT [Automatic Control and Computer Science], no. 1, pp. 9-15, 1987.
3. H. J. Genrich and E. Stankiewich-Weichno, "A dictionary of some basic notions of net theory," Lecture Notes in Computer Science, vol. 84, pp. 519-535, 1980.
4. W. Reisig, "Recursive nets," in: Application and Theory of Petri Nets, First European Workshop, vol. 52, pp. 125-130, Informatik-Facheberichte, 1982.
5. V. E. Kotov, Petri Nets [in Russian], Nauka, Moscow, 1984.
6. M. Silva, "Sur le concept de macroplace et son utilisation pour l'analyse des reseaux de Petri," R. A. I. R. O. Autom./Syst. Anal. and Control, vol. 15, no. 4, pp. 335-345, 1981.
7. A. A. Tal and S. A. Yuditskii, "Hierarchy and parallelism in Petri nets," Avtomatika i telemekhanika, no. 7, pp. 113-122, 1982.
8. K. Jensen, "Coloured Petri nets and the invariant-method," Theoretical Computer Science, vol. 14, no. 3, pp. 317-336, 1981.
9. N. A. Anisimov and V. L. Perchuk, "Representation of exchange protocols by means of sequential automata and Petri nets," Izv. AN SSSR. Tekhn. kibernetika, no. 1, pp. 74-80, 1986.
10. A. P. Ershov, "Scientific fundamentals of conclusive programming," Vestnik AN SSSR, no. 10, pp. 9-19, 1984.
11. C. H. Chow, M. G. Gouda, and S. S. Lam, "On constructing multiphase communication protocols," in: Protocol Specification, Testing and Verification. Proc. IFIP WG 6.1 Fourth Int. Workshop, 1984, pp. 57-68, North-Holland, 1985.

16 October 1986
Revised 13 March 1987