International Journal of Intelligent Information Systems
2013; 2(1) : 1-10
Published online February 20, 2013 (http://www.sciencepublishinggroup.com/j/ijiis)
doi: 10.11648/j. ijiis.20130201.11

Verification of telecommunication protocols based on formal methods

Tkacheva Elena Borisovna, Lubov Demchenko Vasilievna, Saied Halawa Fawaz

Kharkiv National University of Radio and Electronics, Kharkov, Ukraine

Email address:
korov4enko@mail.ru (T. E. Borisovna), gladiy_lv@mail.ru (L. D. Vasilievna), saied.f.h@hotmail.com (S. H. Fawaz)

To cite this article:
Tkacheva Elena Borisovna, Lubov Demchenko Vasilievna, Saied Halawa Fawaz. Verification of Telecommunication Protocols Based on Formal Methods, *International Journal of Intelligent Information Systems.* Vol. 2, No. 1, 2013, pp. 1-10. doi: 10.11648/j.ijiis.20130201.11

Abstract: This article is devoted to the development method for verification and detecting errors that can occur in the operation of protocols for information exchange. The various steps of verification of telecommunication protocols are given in the article; the construction of counterexample, which helps to identify the logical operations that lead to errors in the protocols. Practical implementation of given method is shown on TCP.

Keywords: Verification, Model Checking, E-nets, Formal Grammars, Implementation Model, Specification Model

1. Introduction

Expansion of services set that are provided by info communication technologies and also the progress of information technologies leads to the necessary to develop new network protocols or improve existing ones. Now network protocols have a constant growth in the sphere of requirements of protocols reliability and a list of provided services from users, and increasing of requirements to the time for protocols realization that are implemented by new services from the side of companies at the same time. Thus, the contradiction between requirements and opportunities of design tools, development and deployment of protocols is increasing all time.

As researches show [1, 2] the largest number of errors occur at the stage of gathering requirements and forming protocols' specification. Typically, the protocols' specifications are defined by a subset of natural language. The implementation of the protocol in accordance with these specifications may cause an ambiguous misunderstanding of the requirements, which leads to inconsistent work of protocols' elements or various versions of one protocol. One of methods of elimination this problem is the formal presentation of the specification. Verification is one of the methods that allow determining existence errors in the protocol or service.

The existing methods of protocols' testing have their advantages and disadvantages. In case of applying testing and simulation it is impossible to assess the correctness of protocol behavior in all situations, they only can determine the presence or absence of an error according to a certain scenario. In case of telecommunication protocols verification, the most widely used method is Model Checking method. This method described in [3, 4]. It allows to track the whole set of possible states of the protocol that can identify nonstandard errors.

This method also allows constructing a counterexample that is a variant of the protocol behavior, in which the error can be corrected. However, this method has a significant disadvantage –combinatorial "explosion effect" of the state space that in the case of verification of complex telecommunication protocols makes it impossible to use. Thus there is a need to develop a new formal method for verification of telecommunication protocols, which allows eliminating the effect of the combinatorial explosion.

In this paper, we describe the steps and formal methods (formal specification, analysis and verification) which help to detect errors during the whole life cycle of protocol. The primary contributions of this paper are:

Temporal logics demonstration (Leaner Temporal Logic, LTL and Computational Tree Logic, CTL) which allow describing the consecutive change of the state transitions.

Demonstration modified method for verification which is based on Model Checking and use E-nets as instrument for protocol modeling. Also this method used formal grammas for deeper verification and reduction of the combinatorial explosion (demonstrated in Fig. 2).

Demonstrating the steps of formulation transition firing in E-nets (consider different types of transitions). The developed method of comparing of formal grammar is

based on comparison of languages that describes the behavior of the implementation model and specification model of telecommunication protocol.

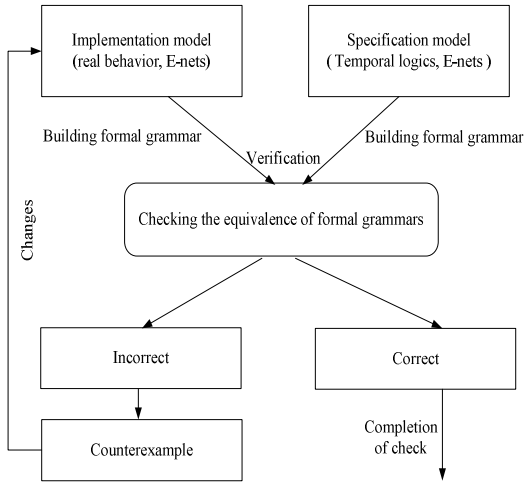

Figure 2. *A modified method of Model checking for verifying telecommunication protocols.*

2. Steps of Detecting Errors on Logic Operation of Telecommunication Protocols

2.1. Formal Specification

The specific of the protocols is that they have a significant number of parallel processes which can potentially interact at any time. Global properties of parallel processes often cannot be formulated in terms of the relationship between inputs and outputs. Temporal logics are applied to facilitate the formal specification of such properties.

Temporal logics allow formulating protocol requirements and describing their basic properties, in sequences of performed events and support the formulation of the protocol behavior changes at any time [5]. The set of requirements, which are submitted by the specification, is defined by the set of atomic utterances. Temporal logics also complemented by temporal operators, which determine an order and a frequency of event occurrence.

Formally, temporal logic is defined as: TL=<A, O, C>, where A - the alphabet of temporal logic; O - the set of temporal operators; C - logical connectives.

Formalization of specification's requirements can be represented by two types of formulas: path formulas $f \models f(a_1 ... a_n)$ - statements true for the lifetime of the process; state formulas $p \models P(a_1)$ - statements are true for a certain state of protocol [6, 7].

Path formulas are built from temporal operators over the states formulas. The state formulas can be true on one state, and the path formulas – during some path. The state formula is a formula of logic language of predicates' calculus over some elements of the program's memory state. Path formulas are constructed from the state formulas, logical and temporal operators.

By means of temporal logics the following classes of temporary properties of telecommunication protocols are marked out:

- Liveliness – a property indicating that the protocol will periodically go to the desired state;
- Security – a property indicating that the protocol is not prone to erratic behavior;
- Correctness – during execution the protocol will enter the desired state;

Often the implementation of the protocol is represented as a mathematical model (usually modeling tools are the varieties of automatons) [8]. The traditional approach of Model Checking method based on complete search and comparison of elements of temporal logic formulas and finding their corresponding position in the implementation protocol model, as well as establishing appropriate linkages between the model states and elements of temporal logic formulas.

If discrepancies between a protocol implementation (the actual behavior, obtained during a model functioning) and its specification are found, then a counterexample is being formed that shows how to eliminate this discrepancy.

In Fig. 1 it is shown the traditional scheme of the Model Checking method.

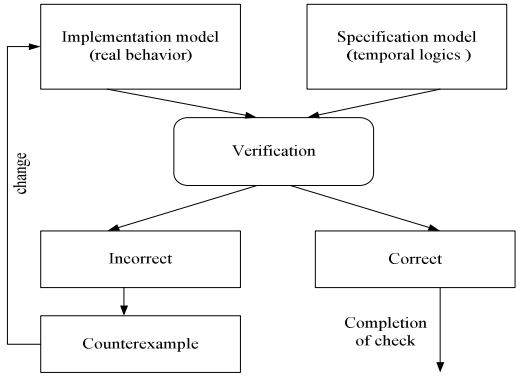

Figure 1. *The traditional scheme of the Model Checking.*

2.2. The Modification of the Model Checking Method

The application of model checking method can be divided into the following steps [3, 4]:

1. Building of mathematical model of the analyzed system and model specification.
2. The formalization of the protocol behavior on the basis of the built model.
3. Presentation of a formal proof of the presence or absence of the specified property in the system.

Within the bounds of the general problem of developing methods for the communication protocols analysis at the stage of requirements forming, specification and implementation of the protocol the E-nets' are selected as a means of modeling [8].

Formally E-network is defined as a bipartite directed

graph: $E = (P, H, L, D, A, M_0)$, where P is a finite set of places, H is a finite set of transitions, L is a direct function of incidence, D is the inverse function of incidence A is a finite set of transition feature set, M_0 is initializes the network.

Thus, the implementation and specification model, represented as E-net model, is the input data for the method of verification which is being developed.

2.2.1. Application of Formal Grammars. Verification on The Basis of Formal Grammars

Formal grammars allow describing the protocol behavior in the form of language words. Word is a state of protocol (according to specification and implementation model), which describe a sequence of protocol state transitions. Thus, the problem of communication protocols verification is reduced to the problem of checking the equivalence of two languages that are based on formal grammars describing their behavior.

Using this method allows to avoid the effect of "combinatorial explosion" of the state space as comparison of two formal grammars is based on sequential checking of equivalence of chains of the languages which describe the specification and implementation of the protocol.

The method of verification based on formal grammar consists in following [9 - 11].

The set of possible behaviors (the chains of language) of implementation and specification model of the protocol is defined by the set of transitions which define the alphabet of formal grammar (Σ). For the specification model the initial state s_0 is defined, which corresponds to the initial statement of temporal logic formula that is true in a given state ($s_0 \models \varphi$).

The final state of the implementation and specification model of the protocol (F) is a key state which must be achieved during functioning of the protocol. It is suggested to use the following modification of the Model Checking method depicted in Fig. 2.

The language is formed from the set of transitions that have been launched during the states transition of the protocol model. For solving the problem of checking the equivalence of the two languages which are generated by a formal grammar, it is necessary to introduce a few statements and definitions.

During checking the equivalence of specification and implementation of the protocol, the following situations can occur:

1. For the specification model only one behavioral chain is built:

$$L(S) = (S_0 h_1 .... h_Z) \quad (1)$$

Where S_0 is the initial state of the model, which is determined by marking of E-net, $h_1 .... h_Z$ is the set of active transitions of the model, $L(S)$ is the behavior language of the model.

2.2.2. Rules for Construction of Formal Grammar for Different Type of Transition in E-Nets

T-transition models the execution of the event, when coming only one condition. To fire T-transition to the lack of label in the output position $p(B) = 0$ and the presence of the label in the input position $p(A) = 1$. The example of T-translation is present in Fig. 3.

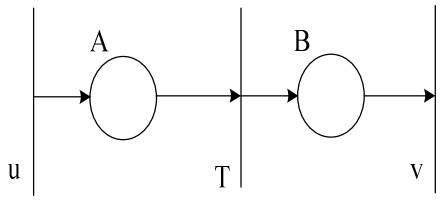

Figure 3. *The structure of T-transition.*

Rule of inference for T-transition has the next form:

$$L(T) = \{uT \mid T \to tv\} = \{utv\}, \quad \{T\} \in N, \{u, v, \mathrm{t}\} \in V_t.$$

F- Transition is used for branching flow conditions or branching flow of transmitted data. For fire of F-transaction needed the lack of label in position $p(B) = 0$ and $p(C) = 0$, and existing label in input position $p(A) = 1$. The example of T-transition is present in Fig. 4.

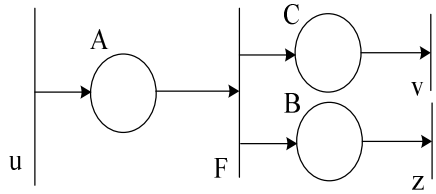

Figure 4. *The structure of F-transition.*

Branching flow can be expressed as the formalism:

$$L(F) = \{uF \mid F \to fzv\} = \{ufzv\}, \ \{F\} \in N, \{u, z, v, \mathrm{f}\} \in V_t.$$

J- transition is used to simulate events that require two conditions at the same time. J-transition is active, only in one case, when both position A and B contains a label, $p(A) = 1$ and $p(B) = 1$ and position C doesn't include label, $p(C) = 0$ (Fig. 5).

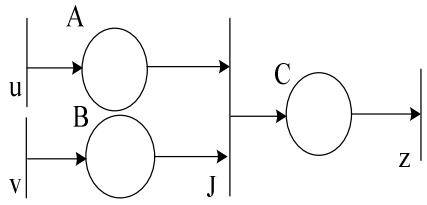

Figure 5. *The structure of J-transition.*

Inference rules for the J-transition have the form:

$$L(J)=\{vvJ \mid J \to jz\} \Rightarrow L(J)=\{uvjz\},$$
$$\{J\} \in \mathrm{N}, \{u,v,z,\mathrm{j}\} \in \mathrm{V_t}.$$

MX- transition sets the direction of labels flow which depending on the value of the predicate $r(S)$ (Fig. 6).

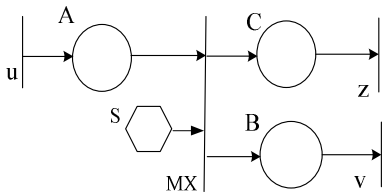

Figure 6. *The structure of MX-transition.*

Chaining is as follows:

$$L(MX)=\left\{\begin{matrix} uSX \mid (u \to X, S \to x \mid X \to z) \Rightarrow \\ \Rightarrow r(S)=0 \wedge (p(C)=0) \to \mathrm{uxv} \end{matrix}\right\} \Rightarrow$$
$$\Rightarrow L(MX)=\{uxv\}$$

or

$$L^{'}(MX)=\left\{\begin{matrix} uSX \mid (u \to X, S \to x \mid X \to a) \\ \Rightarrow r(S)=1 \wedge \mathrm{p(B)}=0) \to \mathrm{uxz} \end{matrix}\right\} \Rightarrow$$
$$\Rightarrow L^{'}(MX)=\{uxz\}$$

Where

$$\{S,X\} \in N, \{C,B,u,v,z,\mathrm{x}\} \in V_t$$

MY- transition is used to model the priority processing of different streams labels (Fig. 7).

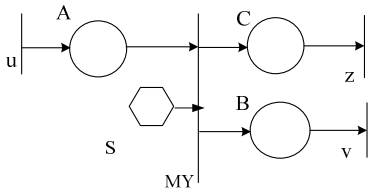

Figure 7. *The structure of MY-transition.*

Chaining is as follows:

$$L(MY)=\left\{ uvSY \left| \begin{matrix} u \to y, v \to y, S \to y, Y \to z \\ \Leftrightarrow r(S)=0 \vee (p(B)=1, p(A)=1, p(C)=0) \to uyz \\ u \to y, v \to y, S \to y, Y \to z \\ \Leftrightarrow r(S)=1 \vee (p(B)=1, p(A)=1, p(C)=0) \to vyz \end{matrix} \right. \right\} \Rightarrow$$
$$\Rightarrow L(MY)=\{uyz \wedge vyz\}$$

Where

$$\{S,Y\} \in N, \{A,B,C,u,y,v,\mathrm{z}\} \in V_t$$

On the basis of the rules of inference of productions for standard types of transitions E-nets formed method parse the entire model of the protocol.

2.2.3. Checking te Equivalence o te Two Languages, Generated by a Formal Grammar

On Fig. 8 shown a fragment of E-net, which specifies one behavioral chain (one variant).

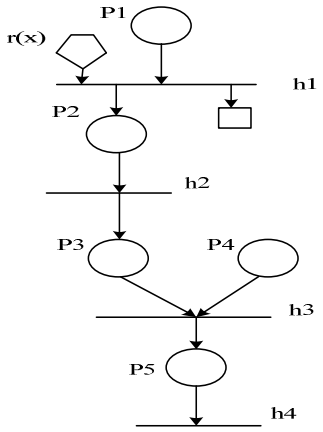

Figure 8*. E-net fragment which corresponds a one variant of state transition.*

The initial state is P_1. The final state is P_5. This language is finite and does not contain cycles. Language that describes the behavior of the given fragment of E-net can be represented as follows:

$$L(S) = (P_1 h_1 h_2 h_3) \quad (2)$$

2. Behavioral chain corresponds to several independent sequences of states transition:

$$L(S) = (S_0 h_1 \ldots \cup S_0 h_i \ldots \cup \ldots h_Z) \quad (3)$$

In Fig. 9 it is shown a fragment of the E-net, for which a few chains may appear. There is a possibility of deadlock formation (transition h_4) in such network. All possible scenarios of model behavior depending on the conditions of triggering of the transition h_1 are following.

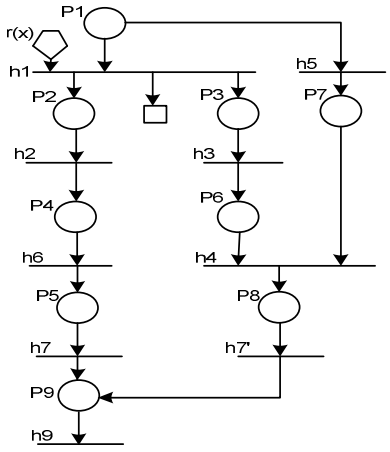

Figure 9. *The E-net fragment that corresponds to several independent sequences of states transitions.*

The initial state is P_1, the final state is P_9. Language

that describes the behavior of the given fragment of E-net can be represented as follows:

$$L(S) = (P_1h_1h_2h_6h_7 \cup P_1h_3 \cup P_1h_5 \cup P_1h_3h_5h_4h'_7) \quad (4)$$

Moreover, the chain P_1h_3 leads to the appearance of a deadlock in the state P_6 if the transition h_5 is not active or vice versa: the chain P_1h_5 leads to the appearance of a deadlock in the state P_7 if the transition h_3 is not active. The transition h_4 is considered to be live (active), if each input position has at least one token.

There are parallel processes interacting with each other (they correspond to the formation of cycles):

$$L(S) = (S_0h_1h_i...(h_j...)^m h_Z). \quad (5)$$

A fragment of the E-net with the possibility of interleave chain has a structure shown in Fig. 10.

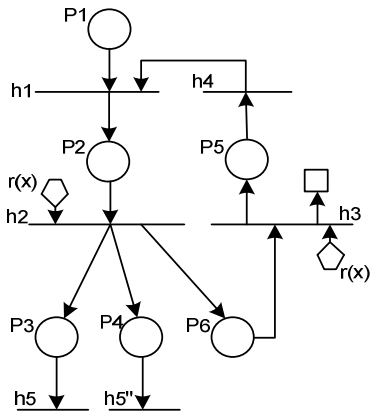

Figure 10. *E-net fragment which corresponds to the formation of cycles.*

The presence of interleaved chains can indicate the formation of loops. Verification of the models in which may appear the loop is the most difficult task. The problem is solved by an additional method of checking – algorithm of double Depth-First Search (DFS) [12]. If interleaved chains $L(S) = (S_0h_1h_i...(h_j...)^m h_Z)$ appear during building of language models, then quantitative assessment of a correspondence between the launch of transition in specification and implementation models. Quantitative ratio can be set by using an additional counter t; it corresponds to the degree of transitions in the model specification language.

In this situation, attributes of predicates have the following meaning: if the number of tokens that hit the position P_4 greater than 3, they will be dropped, and if it is less than 3, then the tokens need to hit the position P_2 again.

Language that describes the behavior of the fragment of E-net (Fig. 10) is represented as follows

$$L(S) = P_1h_1h_2 \cup P_1h_1^3h_2^nh_3^3h_4^2 \quad (6)$$

Thus, the output of the second chain shows that the number of transition activation h_2 does not affect the launch of other transitions, and activation of transition h_3 is allowed only three times. Only in this case, based on the definitions 2 and 3, the chain is allowed. For each of the transitions the counter t has its own value:

$$h_1^3 \mid t=3, h_2^n \mid t=n, h_3^3 \mid t=3, h_4^2 \mid t=2. \quad (7)$$

Using the definitions and statements above, and considering examples of the graph topology of E-nets a method of comparison of the two languages can be formulated. It describes the dynamics of the specification and implementation model behavior of telecommunications protocols:

1. Building a protocol specification language $L(M_S)$. $L(M_S) = (S_0\gamma \mid \gamma \to F)$, where γ - single chain or a set of chains which are generated from the initial state.

2. Performing a step-by-step building of a protocol implementation language:

- determining the current state of the protocol implementation model (in the first step the current state is the initial $S_0(M_R)$);
- determining the set of active transitions $\{h_M\}$;
- forming a launch chain for a set of active transitions: $h_M \to h_{M+1}$.

3. The comparison of built language chains of protocol implementation model with active language chain of protocol specification model.

$$v_i(h_{S_i}, h_{M_i}) = \begin{cases} h_{S_i}, True \mid h_{S_i} \equiv h_{M_i}, \\ \delta(h_{S_i-1}, S_i, h_{S_i}) \equiv \delta(h_{M_i-1}, M_i, h_{M_i}) \\ and \\ 0, False \mid h_{S_i} \neq h_{M_i}, \\ \delta(h_{S_i-1}, S_i, h_{S_i}) \neq \delta(h_{M_i-1}, M_i, h_{M_i}), \end{cases} \quad (8)$$

4. If language chains of protocol implementation model and language chains of protocol specification model are equal then step 2 is repeated with the changing of current state. Otherwise, the counterexample will be build.

2.3. Development of a Method of Forming Counterexamples

The advantage of the suggested method of verification of telecommunication protocols is the ability to form counterexamples. Counterexample can determine the protocol behavior, which can lead to an error [13].

For those situations where a formation of several parallel chains $L(S) = (S_0h_1...\cup S_0h_i...\cup....h_Z)$ is possible, some fragment of the model is returned as a counterexample. It includes an accessibility chain ν and the followers of the last matched transition h_{M_i+1} (it is enough to indicate only the first follower), as well as the state, the occurrence of which is not valid in the given sequence:

$$L(K) = vh_{M_i+1} \cup (s_M \models \psi) \quad (9)$$

To prevent the loops formation in permissive sequences and to check emptiness of protocol implementation model language $L(M)$ the algorithm of double Depth-First Search [12] is used.

This algorithm is designed to find the permissive paths both in specification and implementation models of a protocol and is used by many verifiers, including SPIN verifier and Bogor [14, 15]. In this algorithm, two depth-first searches are interchanged. The first of them can run the second one and the second, in its turn, can either complete the entire algorithm or return control to the first DFS. In this case, the first DFS continues its work. Each DFS uses its own flag to mark the visited states.

The first DFS launches the second one when it is ready to roll back from permissive state (h_j). If during bypassing the second depth-first search enters into the state which is contained in the first DFS stack, then permissive path is obtained. If not, then after terminating bypass the second depth-first search returns control to the first.

Algorithm returns true, if permissive path was found, and false – otherwise. If the algorithm returns true, then it can recover permissive path: in the first DFS stack the path from the initial state h_i to some permissive state h_j is stored. This path is the required suffix β. The second DFS stack stores the path from the state h_i or the initial state of the model to the state h_j which is contained in the first DFS stack.

Then, to complete this path by states that are in the first DFS stack before the state h_j, we get a cycle $h_i \to ...h_j... \to h_i$ that passes through a permissive state h_i, which is the required suffix β. Thus, the permissive path will be obtained:

$$L(K) = v\beta \quad (10)$$

DFS algorithm finds a sequence that is allowed by the E-net if and only if it exists. If the sequence does not exist, the answer false is returned, which may indicate a lack of active transitions in the protocol implementation model.

If the last matching symbol of implementation model does not have any followers, i.e. the state which is determined by the specification does not exist, false will be returned as a counterexample and the sequence will be as follows:

$$L(K) = v \cup \emptyset \quad (11)$$

It should be mentioned that all discrepancies are being searched for each language chain separately. Only after all the chains of the specification model are compared to the chains of implementation model, a final verification result with a positive response (conformance of protocol implementation to its specification) or a counterexample will be returned.

3. Results

As example we concede the process of verification for TCP protocols (connection is established), with use to different operation systems [10, 16].

Model of the connection protocol TCP, showing the desired behavior that is based on the unit E-nets is as follows (Fig. 11 and Fig. 12).

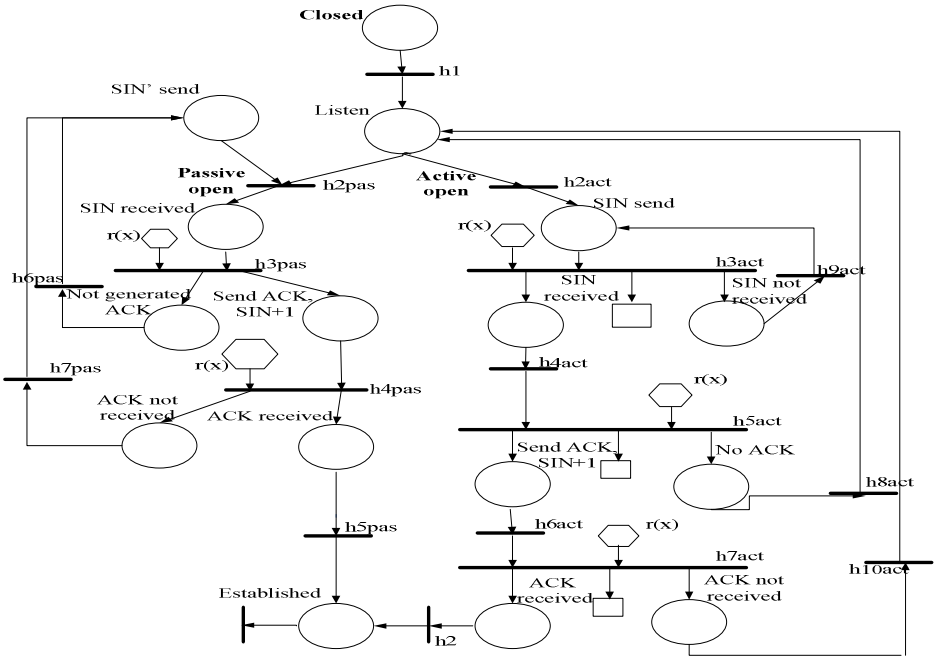

Figure 11. *Specification model for connection process, built on the basis of unit E- net.*

The state Closed – corresponds initial stage of establish connection, Listen – corresponds to the opening of the port to listen for TCP connection requests, SIN' send – corresponds to the fact of the communication with the flag of the SIN of the device (passive connection), SIN' send – corresponds to the formation request SIN device (active compound), SIN' received – corresponds to receiving a message SYN, SIN not received - SYN message is received, Send ACK, SYN +1 - build message ACK (reply message SYN); Not generate ACK - the device does not respond, cannot build message ACK, ACK received - match en receiving a message ACK, ACK not received - ACK message is not received, Established - session is set.

The first step is to establish the verification of the initial state (s_0) corresponding with the initial markup model (M_0), which corresponds to the following markup $M_0 = (1,0,0,0,0,0,0,0,0,0,0,0,0,0,0)$. The initial state is the state of Closed. Only active transition is h_1. The final state is established, which is a no terminal symbol.

Complete description of the protocol behavior scenarios can be represented by the following chain of languages:

$$h_1 \to h_1 h_{2act} \to h_1 h_{2act} h_{3act}$$
$$\to h_1 h_{2act} h_{3act} h_{4act} \cup$$
$$\cup h_1 h_{2act} h_{3act} h_{9act} (r(x) \le 3)$$

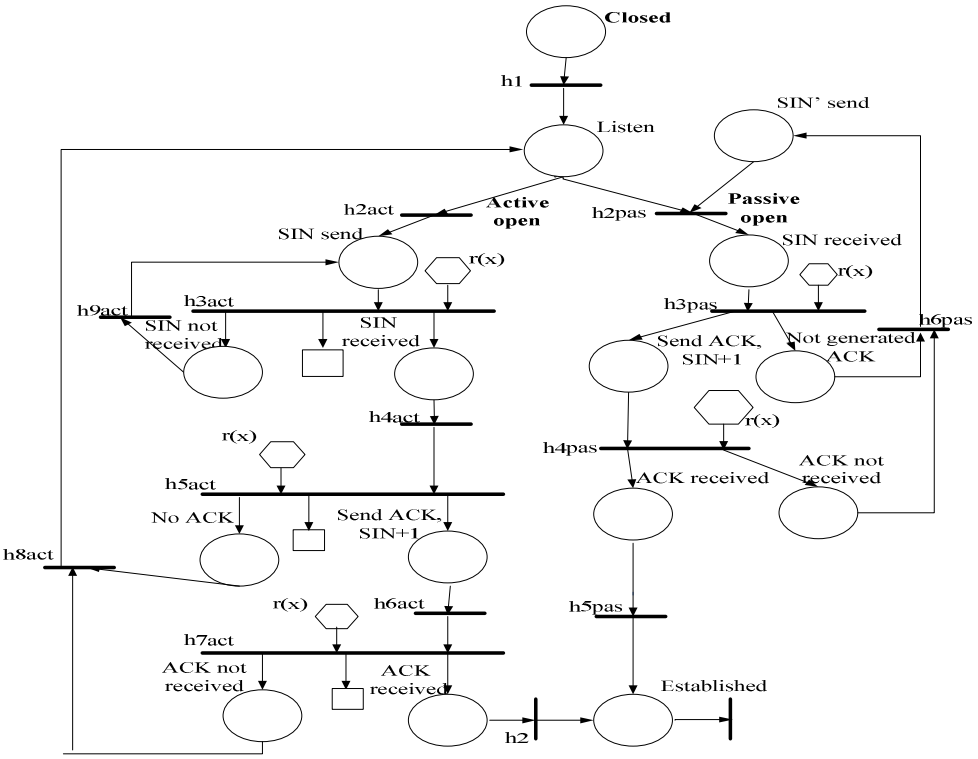

***Figure 12**. Implementation model for connection establishment, built on the basis of the E-machine network.*

As it can be seen from the sequence at start of transition h_{3act} two scenarios are possible connection, we consider each of them separately:

First scenario:

$$h_1 h_{2act} h_{3act} h_{4act} \to h_1 h_{2act} h_{3act} h_{4act} h_{5act} \to$$
$$\to h_1 h_{2act} h_{3act} h_{4act} h_{5act} h_{6act} \cup h_1 h_{2act} h_{3act} h_{4act} h_{5act} h_{8act}$$
$$(r(x) \le 3)$$
$$h_1 h_{2act} h_{3act} h_{9act} (r(x) \le 3) \to$$
$$h_1 h_{2act} h_{3act} h_{9act} (r(x) \le 3) h_{2act} \to$$
$$h_1 h_{2act} h_{3act} h_{9act} (r(x) \le 3) h_{2act} h_{3act} \to$$
$$\to h_1 h_{2act} h_{3act} h_{9act} (r(x) \le 3) h_{2act} h_{3act} h_{4act} \cup$$
$$\cup h_1 h_{2act} h_{3act} h_{9act} (r(x) \le 3)$$
$$h_{2act} h_{3act} h_{9act} (r(x) \le 3)$$
$$\to h_1 h_{2act} h_{3act} h_{9act} (r(x) \le 3)$$
$$h_1 h_{2act} h_{3act} h_{9act} (r(x) \le 3) h_{2act} \to$$
$$\to h_1 h_{2act} h_{3act} h_{9act} (r(x) \le 3)$$
$$h_{2act} h_{3act} h_{9act} (r(x) \le 3) h_{2act} h_{3act} \to$$
$$\to h_1 h_{2act} h_{3act} h_{9act} (r(x) \le 3)$$
$$h_{2act} h_{3act} h_{9act} (r(x) \le 3) h_{2act} \to$$
$$h_{3act} h_{2act} h_{3act} \to$$
$$\to h_1 h_{2act} h_{3act} h_{9act} (r(x) \le 3)$$
$$h_{2act} h_{3act} h_{9act} (r(x) \le 3) h_{2act} \to$$
$$h_{3act} h_{2act} h_{3act} h_{4act} \cup$$
$$\to h_1 h_{2act} h_{3act} h_{9act} (r(x) \le 3)$$
$$h_{2act} h_{3act} h_{9act} (r(x) \le 3) h_{2act} \to$$
$$h_{3act} h_{2act} h_{3act} h_{2act} h_{3act} h_{9act} (r(x) \le 3) \Rightarrow$$
$$\Rightarrow h_1 \underbrace{h_{2act} h_{3act} h_{9act} (r(x) \le 3)}_{3}$$

Sequence

$$h_1 \underbrace{h_{2act} h_{3act} h_{9act} (r(x) \le 3)}_{>3}$$

leads to a dead end, and the rejection of labels (getting into position c). Starting the transition h_{5act} as well as the transition h_{3act} gives rise to two possible scenarios:

$$h_1 h_{2act} h_{3act} h_{4act} h_{5act} h_{6act},$$
$$h_1 h_{2act} h_{3act} h_{4act} h_{5act} h_{8act} (r(x) \le 3).$$

Let us consider each scenario separately.

$$h_1 h_{2act} h_{3act} h_{4act} h_{5act} h_{8act} (r(x) \le 3) \to$$
$$h_1 h_{2act} h_{3act} h_{4act} h_{5act} h_{8act} (r(x) \le 3) h_{2act} \to$$
$$\to h_1 h_{2act} h_{3act} h_{4act} h_{5act} h_{8act} (r(x) \le 3)$$
$$h_{2act} h_{3act}$$
$$\to h_1 h_{2act} h_{3act} h_{4act} h_{5act} h_{8act} (r(x) \le 3)$$
$$h_{2act} h_{3act} h_{4act} \cup$$
$$h_1 h_{2act} h_{3act} h_{4act} h_{5act} h_{8act} (r(x) \le 3)$$
$$h_{2act} h_{3act} h_{9act} (r(x) \le 3) \to \ldots \Rightarrow$$
$$\Rightarrow h_1 h_{2act} h_{3act} h_{4act} h_{5act} h_{8act} (r(x) \le 3)$$
$$h_{2act} h_{3act} h_{9act} (r(x) \le 3) h_{5act} \Rightarrow$$
$$\Rightarrow h_{6act} h_{7act} h_{2act} \cup$$
$$h_1 \underbrace{h_{2act} h_{3act} h_{4act} h_{5act} h_{8act} (r(x) \le 3)}_{3} h_{2act} \Rightarrow$$
$$\Rightarrow h_{3act} h_{9act} (r(x) \le 3) h_{5act} h_{6act}.$$

Sequence leads to the rejection of labels (getting into position c)

$$h_1 \underbrace{h_{2act} h_{3act} h_{4act} h_{5act} h_{8act} (r(x) \le 3)}_{>3}$$

But

$$h_1 h_{2act} h_{3act} h_{4act} h_{5act} h_{8act} (r(x) \le 3) h_{2act} \Rightarrow$$
$$\Rightarrow h_{3act} h_{9act} (r(x) \le 3) h_{5act} h_{6act} h_{7act} h_{2act}$$

to the stage of the connection:

$$h_1 h_{2act} h_{3act} h_{4act} h_{5act} h_{8act}$$
$$(r(x) \le 3) h_{2act} h_{3act} \Rightarrow$$
$$\Rightarrow h_{9act} (r(x) \le 3) h_{5act} h_{6act} h_{7act} h_{2act}$$
$$(Established) Open$$

$$h_1 h_{2act} h_{3act} h_{4act} h_{5act} h_{6act} \to$$
$$h_1 h_{2act} h_{3act} h_{4act} h_{5act} h_{6act} h_{7act} \to$$
$$h_1 h_{2act} h_{3act} h_{4act} h_{5act} h_{6act} h_{7act} h_2 \cup$$
$$\cup h_1 h_{2act} h_{3act} h_{4act} h_{5act} h_{6act} h_{7act} h_{10act}$$
$$(r(x) \le 3);$$
$$h_1 h_{2act} h_{3act} h_{4act} h_{5act} h_{6act} h_{7act} h_2 \to$$
$$\to h_1 h_{2act} h_{3act} h_{4act} h_{5act} h_{6act} h_{7act} h_2;$$
$$(Established) Open$$

$$h_1 h_{2act} h_{3act} h_{4act} h_{5act} h_{6act} h_{7act} h_{10act}$$
$$(r(x) \le 3) \to$$
$$\to h_1 h_{2act} h_{3act} h_{4act} h_{5act} h_{6act} h_{7act} h_{10act}$$
$$(r(x) \le 3) h_2 *$$

Where * is symbolizes about possibility of any of the previous scenarios, $r(x)$ is the number of repeated packages.

Thus, for the model specification condition active open connections meet the following scenarios presented by the language of P-type:

Where *Established* - final state; Closed – initial state.

On the basis of the constructed language of P-type, there are seven basic scenarios connectionless protocol TCP (active open), which include the possibility of accounting retry limit in accordance with the specification.

However, there is the possibility of a third-party request for the establishment of the session in the position SIN' send (p_2), in this situation value of the initial marking is $M_0 = (1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0)$.

$$\begin{aligned}
&L(P) = Closed \to h_1 h_{2act} h_{3act} h_{4act} h_{5act} h_{6act} \to \\
&h_{7act} h_2 (Established) Open \cup \\
&h_1 (h_{2act} h_{3act} h_{9act} (r(x) \le 3))^l h_{5act} h_{6act} h_{7act} h_2 \to \\
&(Established) Open \cup \\
&h_1 (h_{2act} h_{3act} h_{9act} (r(x) \le 3))^l h_{5act} h_{8act} (r(x) \le 3) \to \\
&h_{6act} h_{7act} h_2 (Established) Open \cup \\
&\cup h_1 (h_{2act} h_{3act} h_{9act} (r(x) \le 3))^l \to \\
&h_{5act} h_{8act} (r(x) \le 3) h_{6act} h_{7act} h_{10act} (r(x) \le 3) \to \\
&h_2 (Established) Open \cup \\
&h_1 (h_{2act} h_{3act} h_{4act} h_{5act} h_{8act} (r(x) \le 3))^l h_{6act} \to \\
&h_{7act} h_{10act} (r(x) \le 3) h_2 (Established) Open \cup \\
&h_1 (h_{2act} h_{3act} h_{4act} h_{5act} h_{8act} (r(x) \le 3))^l h_{6act} \to \\
&h_{7act} h_2 (Established) Open \cup
\end{aligned} \tag{11}$$

In this case the behavior of the protocol under the connection can be represented by the following chain:

$$Closed \to h_1 \cup h'_1 \to (h_1 \cup h'_1) h_{2pas} h_{3pas} \to$$
$$(h_1 \cup h'_1) h_{2pas} h_{3pas} h_{4pas} \cup (h_1 \cup h'_1)$$
$$h_{2pas} h_{3pas} h_{6pas} (r(x) \le 3),$$
$$(h_1 \cup h'_1) h_{2pas} h_{3pas} h_{4pas} \to (h_1 \cup h'_1)$$
$$h_{2pas} h_{3pas} h_{4pas} h_{5pas}$$
$$\cup (h_1 \cup h'_1) h_{2pas} h_{3pas} h_{4pas} h_{7pas} (r(x) \le 3),$$
$$(h_1 \cup h'_1) h_{2pas} h_{3pas} h_{4pas} h_{5pas} \to$$
$$(h_1 \cup h'_1) h_{2pas} h_{3pas} h_{4pas} h_{5pas}$$
$$(Established) Open$$

$$(h_1 \cup h'_1) h_{2pas} h_{3pas} h_{6pas} (r(x) \le 3) \to$$
$$(h_1 \cup h'_1) h_{2pas} h_{3pas} h_{6pas} (r(x) \le 3) h_{2pas} \to$$
$$\to (h_1 \cup h'_1) h_{2pas} h_{3pas} h_{6pas}$$
$$(r(x) \le 3) h_{2pas} h_{3pas} \to$$
$$(h_1 \cup h'_1) h_{2pas} h_{3pas} h_{6pas}$$
$$(r(x) \le 3) h_{2pas} h_{3pas} h_{4pas} \cup$$
$$\cup (h_1 \cup h'_1) h_{2pas} h_{3pas} h_{6pas}$$
$$(r(x) \le 3) h_{2pas} h \to$$
$$_{3pas} h_{4pas} h_{6pas} (r(x) \le 3) \Rightarrow$$
$$\Rightarrow (h_1 \cup h'_1)$$
$$\underbrace{h_{2pas} h_{3pas} h_{6pas} (r(x) \le 3)}_{3} h_{4pas},$$

$$
\begin{array}{l}
(h_1 \cup h'_1)h_{2\,pas}h_{3\,pas}h_{4\,pas}h_{7\,pas}(r(x) \le 3) \to \\
(h_1 \cup h'_1)h_{2\,pas}h_{3\,pas}h_{4\,pas}h_{5\,pas}h_{7\,pas} \\
(r(x) \le 3) \Rightarrow \\
\Rightarrow (h_1 \cup h'_1) \\
\underbrace{h_{2\,pas}h_{3\,pas}h_{4\,pas}}_{3}h_{7\,pas}(r(x) \le 3) \cup \\
\cup (h_1 \cup h'_1)h_{2\,pas}h_{3\,pas}h_{4\,pas}h_{6\,pas} \\
(r(x) \le 3 \to \\
)h_{2\,pas}h_{3\,pas}h_{4\,pas}h_{5\,pas} \cup \\
\cup (h_1 \cup h'_1)h_{2\,pas}h_{3\,pas}h_{4\,pas}h_{6\,pas} \\
(r(x) \le 3)h_{2\,pas} \to \\
h_{3\,pas}h_{4\,pas}h_{5\,pas}h_{7\,pas}(r(x) \le 3) \\
h_{2\,pas}h_{3\,pas}h_{4\,pas}h_{5\,pas}
\end{array}
$$

Provided passive opening match following behavioral chain presented by the language of P-type:

$$
\begin{array}{l}
L(P) = Closed \to (h'_1 \cup h_1) \\
h_{2\,pas}h_{3\,pas}h_{4\,pas} \Rightarrow h_{5\,pas} \\
(Established)Open \cup \\
\cup (h'_1 \cup h_1)h_{2\,pas}h_{3\,pas}h_{6\,pas} \\
(r(x) \le 3) \Rightarrow \\
(h_{2\,pas}h_{3\,pas})^l h_{4\,pas}h_{5\,pas} \\
(Established)Open \cup \\
\cup (h'_1 \cup h_1)h_{2\,pas}h_{3\,pas}h_{7\,pas} \\
(r(x) \le 3)h_{2\,pas} \\
\Rightarrow h_{3\,pas}h_{4\,pas}h_{5\,pas} \\
(Established)Open \cup \\
\cup (h'_1 \cup h_1)h_{2\,pas}h_{3\,pas}h_{6\,pas} \\
(r(x) \le 3)(h_{2\,pas}h_{3\,pas})^l h_{4\,pas} \Rightarrow \\
\Rightarrow h_{7\,pas}h_{2\,pas}h_{3\,pas}(h_{4\,pas})^l h_{5\,pas} \\
(Established)Open \mid l \le 3;
\end{array}
\tag{12}
$$

Consider a model of the TCP connection to Unix-like systems (Fig. 11).

According to the method of checking the equivalence given in (8), step through the construction of the language implementation of the protocol, to form a language according to the rules of grammar for the initial state is determined by the implementation model.

In the framework of our implementation of the protocol TCP two connection option investigate: passive open and active open.

Initializes the active connection is opened

$$M_0 = (1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)$$

passive open is possible at the initial marking

$$M_0 = (1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0).$$

In the case of an active open only one active transition presents h_1, $Closed \to h_1$. In passive opening typical transition activity h_1 and h'_1, $Closed \to (h'_1 \cup h_1)$.

Perhaps the formation of several parallel chains

$$L(S) = (S_0 h_1 ... \cup S_0 h_i ... \cup .... h_Z)$$

as a counterexample given fragment of the model, which includes a chain of reach ability v and last matched transition

$$h_{M_i+1}: \; L(K) = vh_{M_i+1} \cup (s_M \models \psi)$$

In the process of checking equivalence of languages that describe the behavior of protocol implementation and specifications models, found the following matching chain:

$$
\begin{array}{l}
v_1 = Closed \\
h_1 h_{2\,act}h_{3\,act}h_{4\,act}h_{5\,act}h_{6\,act}h_{7\,act}h_2 \\
(Established)Open
\end{array}
\tag{13};
$$

$$
\begin{array}{l}
v_2 = Closed \\
(h'_1 \cup h_1)h_{2\,pas}h_{3\,pas}h_{4\,pas}h_{5\,pas} \\
(Established)Open
\end{array}
\tag{14};
$$

$$
\begin{array}{l}
v_3 = Closed \\
h_1(h_{2\,act}h_{3\,act}h_{9\,act}(r(x) \le 3))^l h_{5\,act} \Rightarrow \\
\Rightarrow h_{6\,act}h_{7\,act}h_2 \\
(Established)Open
\end{array}
\tag{15};
$$

$$
\begin{array}{l}
v_4 = Closed \\
(h'_1 \cup h_1)h_{2\,pas}h_{3\,pas}h_{6\,pas}(r(x) \le 3) \\
(h_{2\,pas}h_{3\,pas})^l h_{4\,pas} \Rightarrow \\
\Rightarrow h_{5\,pas} \\
(Established)Open
\end{array}
\tag{16}.
$$

As a counterexample, the following chain:

$$
\begin{array}{l}
L(K)_1 = Closed \\
h_1 h_{2\,act}h_{3\,act}h_{4\,act}h_{5\,act}h_{8\,act}
\end{array}
\tag{17};
$$

$$
\begin{array}{l}
L(K)_2 = Closed \\
h_1 h_{2\,act}h_{3\,act}h_{4\,act}h_{5\,act}h_{6\,act}h_{7\,act}h_{8\,act}
\end{array}
\tag{18};
$$

$$
\begin{array}{l}
L(K)_3 = Closed \\
(h'_1 \cup h_1)h_{2\,pas}h_{3\,pas}h_{6\,pas}
\end{array}
\tag{19};
$$

$$
\begin{array}{l}
L(K)_4 = Closed \\
(h'_1 \cup h_1)h_{2\,pas}h_{3\,pas}h_{4\,pas}h_{6\,pas}.
\end{array}
\tag{20}
$$

Obtained counterexamples (17-20) indicate that transitions h_{8act} and h_{6pas} in the implementation of the protocol does not correspond to requirements of the specification. According to algorithm DFS, for transition firing h_{8act} requires labels in positions p_{11} and p_{13}, that it is impossible. Labels in positions p_5 and p_7 requires for the firing transition h_{6pas}, but that it is impossible in implementation model too.

Thus, this method of comparison chains language models can verify the equivalence of the behavior of the specification and implementation of the protocol.

4. Conclusion

In the article suggested a modified method for verification and detecting the errors that arise in the operation of protocols for information exchange. This method is based on Model Checking method for verifying telecommunication protocols and additionally used the formal grammars to show possible solutions for resolving the issue.

The use of formal grammars also helps to avoid "combinatorial explosion effect" of the state space which constructs the implementation and the specification models.

This effect is achieved through the implementation of sequential equivalence checking chains languages, models describing the behavior specification and implementation of the protocol.

The model specification and implementation protocol, which built using the apparatus E-net, are as input data verification method. Presentation protocol by E-model helps to build one chain behavior of the protocol.

The main advantage of this method compared to the known, is to avoid conflicts between the implementation and specification of the protocol by constructing a counterexample.

To check the efficiency of the developed verification method for equality of languages built on the basis of the rules of formal grammar for suggested verification protocol on TCP with its implementation in Unix systems on establishing connection stage.

References

[1] ISO/IEC 14102:2008. Information technology - Guideline for the evaluation and selection of CASE Tools, 2008.

[2] The Standish Group. The Scope of Software Development Project Failures: The Standish Group. Stanford, 2009, http://www.cs.nmt.edu/~cs328/reading/Standish.pdf.

[3] Jr. Clarke, M. Edmund, and A. Peled, Model Checking, MIT Press, 1999, ISBN 0-262-03270-8.

[4] B. Berard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, and P. Schnoebelen, Systems and Software Verification: Model-Checking Techniques and Tools, 2009. ISBN 3-540-41523-8

[5] C. P. Stirling, “Modal and temporal logics for processes”. LNCS 1043, 1996, pp. 149–237.

[6] J. Bradfield, C. Stirling, “Modal logics and mu-calculi”, Inf.ed.ac.uk

[7] G Nutt, “Evaluation Nets for Computer Systems Performance Analysis”. FJCC, AFIPSPRESS. 1972, pp. 279 – 286.

[8] A. Mironov, “A new method of verification of protocols of data transmission through unreliable medium”, Summer School in Software Engineering and Verification, Moscow, 2011, pp. 261 –276.

[9] N. Chomsky, "Three Models for the Description of Language". IRE Transactions on Information Theory 2 (2). 1956, pp. 113–123. doi:10.1109/TIT.1956.1056813.

[10] E. Korovchenko, “Models and methods for analysis and verification telecommunications protocols based on the E-networks and formal grammars”, Master’s thesis, Kharkov national University of Radio electronics, Oct. 2011.

[11] E. Duravkin, E. Korovchenko, “The formalization of the information exchange protocols behavior provided by models based on E-network”, the problems of telecommunication (e-journal). vol. 1 (3). pp. 28 – 38, 2011: http://pt.journal.kh.ua/2011/1/1/111_duravkin_verification.pdf.

[12] A.T.S. Abu-Jassar, O. Tkachova, “Patterns for reliable Web-services”, The problems of telecommunication (e-journal). vol. 2 (7), pp. 36–42, 2012: http://pt.journal.kh.ua/2012/2/1/122_tkachova_web.pdf.

[13] L. Hoffman, “Talking Model-Checking Technology”, Communications of the ACM, 2008, pp. 110–112.

[14] Formal Grammar: 14th International Conference, FG 2009, Bordeaux, France, July 25-26, 2009.

[15] J. Postel, Transmission control protocol. RFC 793, California, sept. 1981, 85 p.

[16] A. Strunk An algorithm to predict the QoS-Reliability of service compositions. In: SERVICES, 2010, pp. 205–212.