Scenarios Knowledge Base: A Framework for Proactive Coordination of Coalition Operations

Vladimir Gorodetsky and Igor Kotenko

St. Petersburg Institute for informatics and Automation
{gor, ivkote}@mail.iias.spb.su

Abstract. The paper objective is to present a new formal framework for specification of scenario knowledge bases supporting coalition operation dynamic planning, execution and coordination. It is based on formal grammar framework represented is matrix form enriched by attribute component and use of substitution operation allowing hierarchical specification of the scenario world. The proposed framework is demonstrated by a case study dealing with coordinated multi-step attack against computer network performed by hackers.

1 Introduction

Interaction of coalitions involved in a joint operation possesses certain peculiarities noticeably influencing on the coordination strategy of joint behavior to achieve an operation objective. In this application, each coalition has its own semi-autonomous intents, scenarios to achieve these intents, particular resources and constraints. The intents of different coalitions are usually weakly coupled and coordination of their behavior is mostly needed to transform jointly the hostile and unpredictable environment to the state allowing for the coalitions achieving their intermediate and ultimate goals. In the most cases this assumes partial ordering of coalition activities. Other peculiarity is that the coalition goals both intermediate and ultimate, can be achieved by use of multiple and flexible scenarios. Moreover, coalition intents can change depending on both evolution of intents of operation as a whole and state and response of environment leading to modification of scenarios and coordination.

Multi-agent system theory, which is now considered as a basis for coalition operation modeling, simulation and control, the protocol is assumed as the main mean of collective behavior coordination. However, the nature and peculiarities of coalition operations practically exclude use of protocol as coordination framework because as a rule coalitions are weakly coupled entities, which need some kind of "soft" coordination. Indeed, coalitions behavior is much more complex, sophisticated and less deterministic, what makes it difficult to coordinate coalition behavior by use of a protocol. Coordination mode here has to leave to the coalitions much more flexibility in both selection of a concrete scenario leading to joint intent achievement and real-time scheduling of the scenario execution. As a rule, coalition operations are executed in hostile environment with certain degree of unpredictability. Due to these circumstances, coalitions are not able to in advance select scenario leading to a predetermined intermediate or ultimate goal. This means that coalition behavior scenario has to be dynamically constructed in real-time and step-by-step modes with feedback from environment and other coalitions.

The task of coalition coordination is better match problem statement peculiar to traditionally considered in BDI (*Belief-Desire-Intention*) approach. However, to practically use this approach, it is necessary to provide each entity (agent, coalition) with sophisticated knowledge bases, trustworthy and predictive knowledge about environment and other entities acting in common environment, and also powerful and efficient reasoning mechanisms. As a rule, modal and temporal extensions of the first order logic used in BDI approaches, although possessing the needed expressive power for knowledge and believes representation, do not propose computationally efficient reasoning mechanisms ([2], [6], [9], etc.). Perhaps, this kind of frameworks has certain perspective but not in the nearest future.

This paper introduces a formal framework for specification of scenario knowledge bases integrated with an efficient reasoning mechanism. The basic ideas of this framework were firstly proposed in [8] and then developed in [4]. The framework is based on use of a special representation form of the Context Free (CF) grammars ([1], [7]) enriched by attribute component. Under some assumptions and simplifications, within the class of application in question, the framework allows for different coalitions to dynamically (step-by-step) construct coordinated scenarios of their joint behavior depending on their current intents, coalition states achieved and response from the potentially hostile and/or unpredictable environment. The main assumption adopted is that the set of activities used in different scenarios are partially ordered and coordination is mainly destined to meet partial order relation imposed on the different coalition actions. In other words, a coalition can perform certain action if and only if certain intermediate or/and ultimate goals of other coalitions have already been achieved. Though the framework possesses certain limitations[1] of expressiveness of scenario representation language, it allows efficient solving a broad range of tasks of dynamic evolving and coordination of joint coalitions behaviors. However, due to the fact that coalition operations, along with computer support, are directed and managed by human intellects with a lot of shared non-formalized contexts and knowledge, coordination within coalition operations needs much less expressiveness of the formal framework specifying this kind of coordination as compared with the case if such coordination is performed within agents community, which shared knowledge and context are much poorer.

The rest of the paper is organized as follows. Section 2 introduces the basic components of scenario knowledge base and structure given over them at conceptual level. In section 3 the introduced conceptual model is demonstrated by example for an application dealing with coordinated distributed attacks of hackers against computer network. In section 4 scenario knowledge base formal specification language using the introduced framework is presented. Section 5 outlines the general idea of reasoning mechanism destined for dynamic inference of coalitions coordinated scenarios in proactive manner and demonstrates this mechanism by example Conclusion discusses the proposed framework and future works.

2. Basic Components of Scenario Knowledge Base Model

To be capable to dynamically compute scenarios, all the coalitions have to possess knowledge and beliefs that allow them to realize goal–directed (proactive) behavior taking into account constraints, limited resources and opposition of the environment.

[1] Like any other formal framework

Specification of the coalition knowledge and believes is made in terms of *scenario knowledge* introduced below. Scenario knowledge represents formally the set of admissible sequences of actions of coalitions participating in joint operation. This knowledge, while represented formally, is used for dynamic inference of admissible course of coalitions coordinated actions driven by current state of scenario knowledge base, goals achieved and response of the environment. Scenario knowledge is specified in terms of the following basic hierarchically ordered notions: *Simple Behavior model*, *Behavior Model*, *Scenario Knowledge base* and *Mechanism of Dynamic Scenario Inference*. Let us introduce these notions.

2.1 Simple behavior model

Conceptually, *Simple Behavior Model* (*SBM*) comprises a set of *input* $\{X_i\}_{i=1}^{n}$, *activity (transformation operator)* B, *precondition C(B)* and *output* Y. Input and output are also called below as *"goals"* or *"variables"*. In particular applications input and output can be interpreted differently and can be represented by particular data structures. As applied to coalition operations, a variable is interpreted as a goal achieved or to be achieved. Its identifier denotes both a goal and associated data structure representing the goal specification. The latter contains special binary attribute indicating the status of the goal (achieved or not achieved). The examples of coalition goals are to achieve certain geographical position (data structure specifies the coalition resources at given time), to obtain certain information about environment, readiness of a hospital to acceptance of victims, etc.

Activity (transformation) is interpreted as an atomic step of coalition operation destined for achievement of a new goal. It is associated with the expenditure of resources and time. Examples are the *process* of a hospital deployment, a negotiation *process*, hackers' attack in progress, etc. Each activity is assigned a type, $Type(B) \in \{Act,\ SS\}$. If it is completely defined how to execute an activity, it is assigned the type *Act* having sense of *atomic behavior*. If an activity may be executed according to multiple scenarios and choice is to made dynamically it is assigned type *SS* ("*Sub–Scenario*"). The latter assumes that the respective action is multi–step and also is the subject of choice. For example, if an action represents a joint coalition activity that action of type *SS* represents activities of a coalition. Thus, activity of type *Act* is defined completely, whereas one of type *SS* corresponds to, possibly, a set of sub–scenarios. It is worthy to note that use of activities of type *SS* allows hierarchical specification of scenario knowledge. This aspect is explained below.

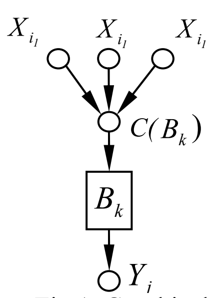

Fig.1. Graphical explanation of *Simple Behavior Model*

Precondition *C(B)* of activity *B* specifies conditions under whose the activity may be executed. The examples of preconditions are certain weather or day period when activity is to be executed for potential its success (photography is to be done at summer time). Within a case study considered below, for potential success of certain attacks against computer network preconditions can be such as "*Network firewall (host) passes through udp-packets* " or "*Linux platform is installed*". In other words, preconditions represent constraints that are mostly out of a coalition control. As a rule, preconditions represent unavailable information about environment. For example, if hacker does not know what kind of operating system is installed in the host subjected an attack, and he performs an attack that is effective if only OS *Widows NT* is installed in the host, then in case the OS *Linux* is actually installed in it, the attack is fail.[1]. Thus, formally *SBM* is a four–tuple

$$SBM = < \{X_i\}_{i=1}^{n},\ B,\ C(B),\ Y >, \qquad (1)$$

i.e. <*inputs, activity, preconditions, output*>, where inputs correspond to the goals that have already been achieved, whereas output is the goal coalition aims to achieve. Fig.1 demonstrates the notion of *SBM* graphically.

The union of inputs $\{X_i\}_{i=1}^{n}$, output Y are below called as *informational component* of *SBM* whereas precondition *C* and activity *B* –its *behavioral* component.

2.2 Behavior Model

Behavior Model (BM) is defined as a structured set of *SBMs. BM* can represent scenario knowledge of coalitions concerning, for instance, particular coalition operation. As a rule, coalition operation can be developing due to different scenarios aiming at the same ultimate goals (intents). On the other hand, coalitions can participate in different joint operations with various initial states and ultimate goals. Thus, if coalitions can execute *N* joint operations[2] then coalition scenario knowledge are composed of *N* behavior models BM_k, *k=1,...,N*, such that $BM_k = \{SBM_r\}_{r=1}^{s_r}$ with given structure over the last set. Let us introduce this structure.

It is given by two relations between so called behavioral and informational components of the set $\{SBM_r\}_{r=1}^{s_r}$ ant that:

–*Behavioral* component of BM_k, *k=1,...,N*, is defined as union of behavioral components of $\{SBM_r\}_{r=1}^{s_r}$ composing BM_k. Let us denote it as $\boldsymbol{B}_k$.

–*Informational* component of BM_k, *k=1,...,N*, is defined as union of informational components constituting behavior model BM_k. Let us denote it as $\boldsymbol{X}_k$. It includes the complete set of coalition operation goals and data structures associated with them.

Let us outline how *SBMs* are structured within BM_k. This structure is determined by two relations. The first of them*, immediate successor relation* U_{BX} is given over pairs of Cartesian production $\boldsymbol{B}_k \times \boldsymbol{X}_k$. In it, for every activity $B \in \boldsymbol{B}_k$ *the immediate successor* is one and only one element of the informational component $\boldsymbol{X}_k$ of BM_k which is conceptually interpreted as the goal of an activity *B*. The second relation, *immediate predecessor relation*, V_{XB}, is defined over the Cartesian production $\boldsymbol{X}_k \times \boldsymbol{B}_k$. In it, every pair $< X_i, B_j >$ is interpreted as "variable X_i is an input of activity B_j".

[1] Sometimes it is ambiguous whether certain entity is to be qualified as variable or as precondition. A decision mainly depends on the task context.

[2] In different operations, coalitions can use the identical activities represented by the same *SBM*s. But in different operations the identical *SBMs* can be structured in various ways. An analogy with Kripke model of possible worlds could be here pertinent.

Thus, behavior model BM_k is defined as four–tuple

$$BM_k = < \{B_j\}_k, \{X_i\}_i, U_{BX}, V_{XB} > , \tag{2}$$

2.3 Scenario Knowledge Base

The last important notion of scenario knowledge of coalitions is *substitution operation*. The necessity to use this operation is caused by the fact that behavioral component of BM_k can contain activities of two types, Act and SS, at that activity of type SS does not determine any particular procedure but only refers to a scenario specifying it in terms of particular activities, possibly, of both types Act and SS.

Let $SBM{=}<\{X_i\}_{i=1}^{n},\ B,\ Y>$ and $BM_k = < \{B_j\}_k, \{Z_i\}_{i=1}^{R}, U_{BZ}, V_{ZB} > -$ a Behavior Model, $SBM \in BM_k$. Substitution of an activity of type SS in SBM by an activity of type Act or by the name of other BM is called substitution operation. Such an operation either concretize activity of type SS, or represents it in terms of a new behavior model $BM \in \{BM_k\}$, which is a scenario. Thus, substitution operation extends the scenario model via allowing for specification of scenario hierarchy.

Scenario Knowledge Base (SKB) of Coalition operation is defined as the set of behavior models (3) with the given set of substitution operations. Let us note that this definition determines composition, destinations and structure of SKB conceptually. Its formal specification language is considered later in section 4.

2.4 Coalition Operation State

An important notion associated with coalition operation specification is its state. *State of a Coalition Operation* at a given time is defined as the set of goals achieved by its coalitions to the given time. In the same way the *State of a Coalition* is defined. If to compare notion of "*Knowledge base*" and above introduced notion "*Scenario Knowledge Base*", it can be mentioned that the notion "goals achieved" in the last case is analogical to the notion "facts" in the former one and the "state of a coalition operation" is an analogy of the "set of all the facts" (input and inferred).

However, the state of a coalition (coalition operation) represents not only current coalition knowledge about its own status, but also its updated believes about environment. Together with current state of SKB (see below what it is) state of coalition (coalition operation) completely determines coalition current state of knowledge about itself and about environment.

The introduced conceptual model of SKB is explained in the next section by the developed case study from computer security domain.

3 Case Study: Scenario Knowledge Base of Hackers

Let us consider an example representing a fragment of the SKB of hackers realizing distributed coordinated attack against computer network. It is assumed that this SKB is used by hackers to dynamically form the scenario of joint attack. The necessity to use dynamic design of scenario is caused by the fact that each next step of hackers depends on the reaction of the "counterparty" on the previously implemented steps.

The Tables 1, 2 and 3 introduce the terminology used in the *SKB* description and denotations of particular entities specifying scenario knowledge. In particular, Tab.1 presents the list of identifiers of input and output variables (intermediate and ultimate goals). Tab.2 presents the list of attack identifiers and minimal explanations of the attacks essences. The attacks correspond to that what is called above activities. It should be noted that all the attacks correspond to the activities of type *SS*, and have to be further concretized by the scenarios of attack realization, which also are represented in terms of attribute grammars [5]. Tab.3 represents the list of preconditions used in the model of hackers *SKB* unknown for hackers. Let us note that he developed case study includes more than 100 different kinds of attacks.

Table 1. List of identifiers of variables of the domain "Computer Networks Attacks"

Identifier	Goals of attacks	Identifier	Goals of attacks
X1	<IP-address of network (or host)>	X21	<Registry is accessible>
X2	<IP-addresses of active hosts>	X24	<e-mail address is known>
X3	<Active ports of a host>	X25	<Password is known >
X4	<The type of OS is known >	X31	Connection is closed
X14	<Shared Resources are available>	X32	<Admin (root) Password is known>
X15	<Users' Identifiers are known>	X37	<File(s) is (are) deleted>
X16	<Security identifier (SID) is known>	X39	<The logs are cleared>
X19	<Host resources are accessible> (Password to access resources)	X41	<Back doors are created>

Table 2. Names of attacks used in the example

Class of attack	Name and content of the attack instance	Identifier	Input	Output
IH: Identification of Hosts	DC: Network Ping Sweeps	SS1	X1	X2
	STIH: TCP connect scan	SS2	X1	X2
IS: Identification of Services	SS: TCP SYN scan	SS5	X2	X3
	SX: TCP Xmas Tree scan	SS7	X2	X3
	SU: :UDP scan	SS9	X2	X3
IO: Identification of OS of the host	IF: ICMP message quoting	SS25	X2	X4
	IDOS: Examination of response for DoS attacks Ping of Death, WinNuke, Teardrop, Land for detection of a Windows OS type	SS28	X2	X4
	RMT: Enumerating NetBIOS Shares with Rmtshar	SS37	X2	X14
	LEG: Enumerating NetBIOS Shares with Legion	SS45	X2	X14
UE: Users and groups	FUE: Finger Users Enumeration	SS52	X2	X15

	ISU: Identifying of Security identifier of User	SS49	X2	X16
	IAS: Identifying Account with sid2user	SS50	X2, X16	X15
GAR: Getting Access to Resources	UDG: User Data Guessing for Access to the Registry	SS63	X2, X14, X15	X21
	MP: Mailing password and access to a host	SS75	X2, X15	X25
	CC: Connection Closing	SS81	X2, X15	X31
EP: Escalating Privileges	UKE: Use of Known Exploits (Ls_messages, getadmin, sechole, etc.)	SS84	X2, X15, X25	X32
IVR: Integrity violation	DFR Delete File(s) Realization	SS89	X2, X15, X31	X37
CT: Covering Tracks	CL: Clearing Logs	SS98	X2, X15, X31	X39
CBD: Creating Back Doors	CRUA: Creating Rogue User Accounts	SS100	X2, X15, X31	X41

Table 3. List of conditions

Name	Condition	Name	Condition
C1	Network firewall (host) passes through icmp-packets echo request	C18	Access to the shared resources (directory) with Record Permission
C2	Network firewall (host) passes through tcp-packets	C21	Personal Web server (PWS)
C3	Network firewall (host) passes through udp-packets	C26	IP-address of an attacking Host is written to the File .rhost
C5	Network (host) firewall passes through tcp/ip-packets	C27	Trojan Horse for password stealing is implanted
C8	Windows (NetBIOS)	C24	Windows
C9	Connection “null sessions” (NS)	C25	Linux or Unix
C14	Microsoft Remote Registry Service (RR) is accessible	C28	TCP and UDP ports 135 - 139 and also 445 (for Win 2000) are open &
C16	Shared files and printers (SFP)	C29	Service finger (port 79) is active and accessible

Fig.2 presents a fragment of the developed case study introduced in Tab.1-Tab.3 in structured according to the relationships U_{BX} and V_{XB} described in previous section.

4 Formal Specification of Scenario Knowledge Bases

Let us consider briefly formal framework for *SKB* specification and demonstrate it by example extracted from the *SKB* depicted in Fig.2. It is given below in Fig.3. This

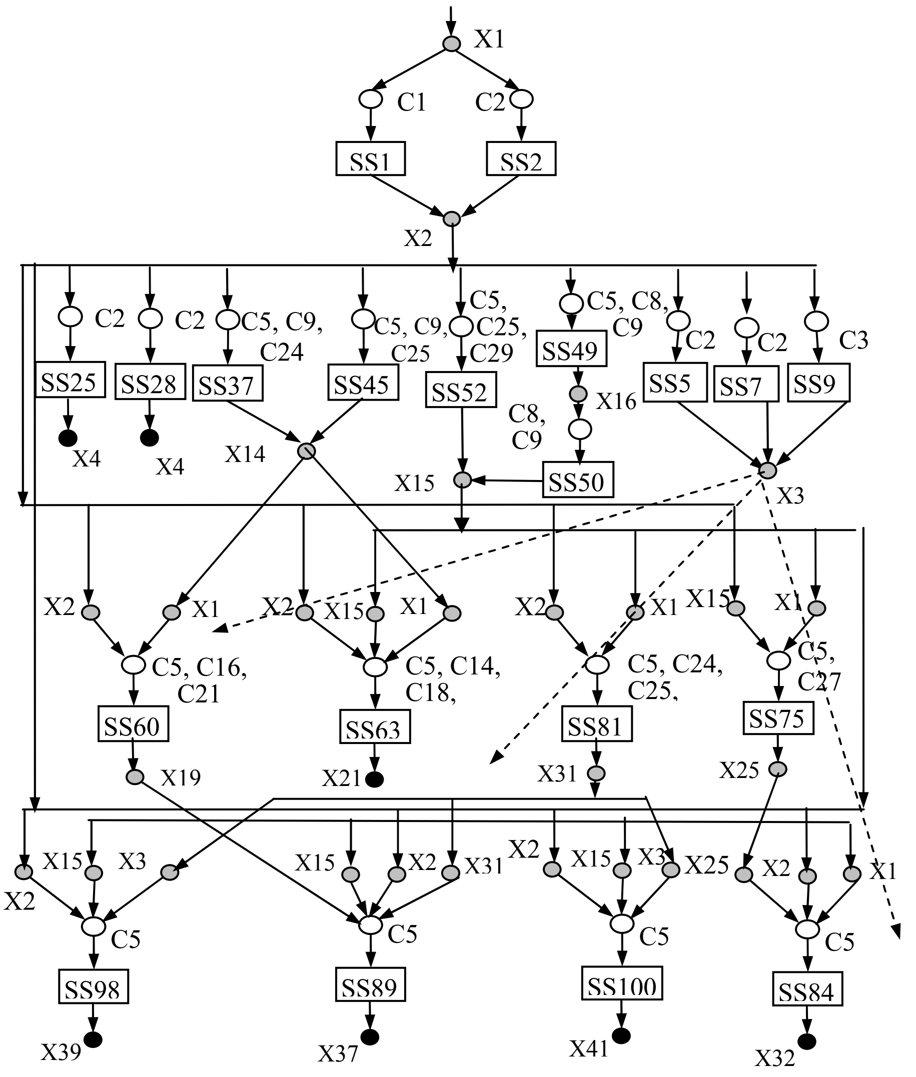

Fig.2. Fragment of Behavior Model for the domain " Computer network attacks ".
Legend: ● –goals, ○–conditions, ●–ultimate goals, $\boxed{\text{SS89}}$–activities (processes).

framework specifies *SKB* in terms of attribute formal grammars with Context Free (CF) kernel, at that each Behavior Model is represented in it as follows:

$$G(BM) = \langle V_A, V_T, S, R \rangle \qquad (3)$$

(subscript indicated the number of *BM* is omitted), where $G(BM)$ –grammar, V_A – the set of nonterminals, V_T –the set of terminal symbols, S –the set of initial nonterminals (grammar axioms), and R – the set of attributed productions.

The first and second members of (4) are interpreted as follows: $V_A = \{X_i\}$ –the set of identifiers of inputs and outputs (goals) of the *SKB*, and $V_T = \{x_i\}$ -the same variables if they are assigned particular values. The axioms sense is explained below.

Let us explain attributed productions. Let us consider SBM_s of the *Behavior Model BM*, and represent it in the form of a production with attribute component:

$$X_i \longleftarrow < pre-condition\ C_s > [B_s]\boldsymbol{X}_s \qquad (4)$$

where X_i–nonterminal put in correspondence to the output of the *Behavior model* B_s, $[B_s]$–attribute component of the production (the name of activity B_s of SBM_s), *C*–precondition and $\boldsymbol{X}_s$ – the vector of inputs of the activity B_s, whose components are ordered in the manner of increasing of its subscripts.

The union of attributed productions like (5) for all $SBM_s \in BM$ constitutes the set R of attribute grammar $G(BM)$ presented in (4).

Let us explain the formal grammar specification of *SKB* by small fragment of *SKB* presented in Fig.3 that is extracted from *SKB* given in Fig.2. While follow the above explanations of how attribute formal grammar-based specification of *SKB* is built, formal specification of *SKB* fragment (Fig.3) is is follows:

$V_A = \{X_2, X_{14}, X_{15}, X_{16}, X_{21}, X_{25}, X_{32}.\}$, $V_T = \{x_2, x_{14}, x_{15}, x_{16}, x_{21}, x_{25}, x_{32}.\}$

$R(1) = \{X_2 \leftarrow X_2; X_{14} \leftarrow [SS37]X_2; X_{14} \leftarrow [SS45]X_2; X_{21} \leftarrow [SS63]X_2X_{14}X_{15};\}$

$R(2) = \{X_2 \leftarrow X_2; XX_{15} \leftarrow [SS47]X_2; X_{15} \leftarrow [SS50]X_{16};$
$X_{16} \leftarrow [SS49]X_2;; X_{25} \leftarrow [SS75]X_2X_{15}; X_{32} \leftarrow [SS84]X_2X_{15}X_{25};\}$

where $R(1)$ and $R(2)$–the sets of productions specifying the scenario knowledge bases of the Hacker 1 and 2 respectively (hackers are considered here as coalitions).

Note. Knowing of X_3 ("list of active ports of the host") is necessary to success of implementation of all the attacks presented in Fig.3. To simplify the expressions of productions in the sets $R(1)$ and $R(2)$ hereinafter X_3 is ignored.

In addition to $R(1)$ and $R(2)$, the following production can be included:

$$x_i \leftarrow S_i,\ i=1,...n, \qquad (5)$$

at that they are included if and only if the goals corresponding to the respective variables have already been achieved by hackers.

Let us transform the introduced formal grammar-based representation of *SKB* in matrix form [7]. Such a representation consists of two components. The first one (denoted below as $\boldsymbol{X}$) is the vector whose *i-th* component is the union of all the chains inferable from formal grammar $G(BM)$ (4) in case if the *i-th* axiom S_i is used as the grammar axiom. The second component is a matrix $\boldsymbol{M}(\boldsymbol{X})$ that in our case (for attribute grammar $G(BM)$) is constructed as follows. It comprises *n* arrows and *n* columns like vector $\boldsymbol{X}$ (In the example Fig.3. $n=4$ for the left *SKB* and $n=5$ for the right one). The matrix represents the current state of coalition knowledge. The initial state of matrix $\boldsymbol{M}$, $\boldsymbol{M}(\boldsymbol{X}^{(0)})$, is formed as follows. In its cell with indexes i,j the chain $[B_k]X_{i_1}X_{i_2}\dots X_{i_k}$ is placed if production $X_i \rightarrow [B_k]X_{i_1}X_{i_2}...X_{i_k}X_j$ exists in grammar $G(BM)$. In other words, this chain is placed in the cell (i,j) if there exists a production with nonterminal X_i in its left part and with terminal X_j in its last position (if variables X_s of the activity are ordered in subscript *s* increasing mode). If several productions containing the same symbol X_j are present in the last position then all they are placed in the same cell joined by symbol "+" interpreted below as union of chains. Otherwise, in this cell the symbol of annihilating chain "∅" is placed.

Thus, in terms of the introduced vector $\boldsymbol{X}$ and grammar matrix $\boldsymbol{M(X)}$ the complete set of the chains that are inferable in formal grammar (4) can be represented as the result of matrix iterations as follows [7]:

$$\boldsymbol{X}^{(0)} \leftarrow \boldsymbol{S},\ \boldsymbol{X}^{k+1} = \boldsymbol{M}(\boldsymbol{X}^k)*\boldsymbol{X}^k + \boldsymbol{X}^k,\ for\ \ k=1,\ 2,\ \ldots, \tag{6}$$

where the symbol "*" is interpreted as chains concatenation.

The remaining question is about how initial vector $\boldsymbol{X}^{(0)}$ is built. Formally, it is built by use of production (6), which is always used as first because its components are axioms. Instantiation of the components of vector $\boldsymbol{X}^{(0)}$ is carried out as follows. If the goal x_i is not "achieved" in the initial state than the production $\varnothing \leftarrow S_i$ is used, otherwise $x_i \leftarrow S_i$. In the considered case study all the position of vector $\boldsymbol{X}^{(0)}$ corresponding to known information about the host to be attacked are assigned as x_i.

Let us demonstrate how to build the matrix form of attribute grammar of *SKB* based on the grammar built above for *SKB* presented in Fig.3.

Formal specification of SKB of the hacker 1 in the initial state:

$$\boldsymbol{X}^{(0)}(1)=\begin{bmatrix} x_2 \\ \varnothing \\ \varnothing \\ \varnothing \end{bmatrix},\ \boldsymbol{M}(\boldsymbol{X}^{(0)}(1) = \begin{bmatrix} \varnothing & \varnothing & \varnothing & \varnothing \\ [SS37]+[SS45] & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & [SS63]X_2X_{14} & \varnothing \end{bmatrix} \tag{7}$$

Formal specification of SKB of the hacker 2 in the initial state:

$$\boldsymbol{X}^{(0)}(2)=\begin{bmatrix} \varnothing \\ \varnothing \\ \varnothing \\ \varnothing \\ \varnothing \end{bmatrix},\ \boldsymbol{M}(\boldsymbol{X}^{(0)}(2))=\begin{bmatrix} \varnothing & \varnothing & \varnothing & \varnothing & \varnothing \\ [SS52] & \varnothing & [SS50] & \varnothing & \varnothing \\ [SS49] & \varnothing & \varnothing & \varnothing & \varnothing \\ \varnothing & [SS75]X_2 & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & [SS84]X_2X_{15} & \varnothing \end{bmatrix} \tag{8}$$

5. The Basic Idea of Inference Mechanism

Reasoning in *SKB* aims at solution, for each coalition, the following task:

Initial state: The set of goals $\{x_i\}$ already achieved represented by vector $\boldsymbol{X}^{(0)}$;

The set of goals $\{x_{i_1}, x_{i_2}, \ldots x_{i_r}\}$ to be achieved by coalition(s).

The task: To dynamically (step by step) compute potential scenarios and points of coordination leading to achievement of the goals set $\{x_{i_1}, x_{i_2}, \ldots x_{i_r}\}$.

Therefore, the objective of the reasoning is to dynamically construct potential strategy of coordinated coalition operation. It is assumed that in the initial state the coalition can exchange information about their states and the same is true during operation execution. At that, reasoning results have also to indicate to the coalitions when and how they need to coordinate their activities. All the steps of reasoning are analogical. The difference is only in the initial states of coalition(s). We below don't discuss how the information exchange is organized because this depends mostly on software design issue. For example, if multi-agent architecture is used for software implementation then the above problem can be solved by use of shared ontology.

It is assumed that in each step of reasoning and for each coalition the supporting computer system "knows" current state of this coalition (the set of goals achieved) and possesses the necessary information concerning some other coalitions. The results of reasoning step are presented to the coalition in form of the set of admissible activities it can perform "to go" towards the ultimate goal(s). The coalition decides what of them and in which order to execute and starts to act. Due to hostile or/and unpredictable reaction of the environment, certain or all proposed activities can fail. After completion of activities inferred in *SKB*, the coalition updates (or not update) its state by adding new goals achieved by it or by certain other coalitions. The system updates current state of its *SKB* and the process it repeated with new "*initial state*".

In the considered case study each hacker's state is represented in terms of information available to hackers about the computer network or host under attack. This information can be obtained either (1) in previous steps of attack, or (2) received from other hackers, or (3) received by social engineering (reconnaissance) method.

In considered simplified example coordinated attack is performed by two hackers. Each of them possesses a part of scenario knowledge, which is represented in his *SKB*. The ultimate goals of coordinated attack are to achieve x_{21} ("*Registry is accessible*") и x_{32} ("*<Admin (root) Password> is known*"). It is assumed that initial state of the first hacker is x_2 (*<IP-addresses of active hosts> are known*). It is assumed that hackers don't know how to detect the operating system (OS) type installed on the host, but some attacks can be only successful if a definite OS (*Windows, UNIX, or Linux*) is installed (Fig.3 and Tab.3). These facts are reflected in attack preconditions. Since hackers do not initially know the type of OS installed, certain attacks can fail. This is the case when environment (network security software) undesirably reacts on the hackers' activity. In such cases hackers are forced to search for new continuation of the attack. *SKB* helps him to dynamically find such continuations if any and also supports hackers' information exchange to share known information when necessary.

Let us describe the first step of each of two hackers having initial state $\boldsymbol{X}^{(0)}$. Hackers' *SKB* are presented in Fig.3. Inference (reasoning) mechanism is realized as matrix iteration (7) for matrix grammars $\boldsymbol{M}(\boldsymbol{X}^{(0)}(1)$ and $\boldsymbol{M}(\boldsymbol{X}^{(0)}(2))$ presented by expressions (8) and (9) respectively. These matrices together with vectors of initial states $\boldsymbol{X}^{(0)}(1)$ и $\boldsymbol{X}^{(0)}(2)$ specify initial state of *SKB*s [7]. Let us demonstrate the first steps of reasoning by both hackers and explain some specificity of the inference mechanism implementing reasoning.

> *Note. The subsequent description of inference mechanism is given in a simplified form of "data driven forward chaining reasoning", when no special means to improve its efficiency are used. Fortunately there exist a number of "tricks" allowing noticeably increase the inference efficiency. Example is alternating "data driven forward chaining reasoning" with "goal driven backward chaining reasoning", what results in good*

efficiency. We don't discuss below the details of rational organization of reasoning mechanism because this aspect is not the main subject of the paper.

Let us assume that for the host subjected to hackers' attack the following preconditions are held: C5, C9, C25, C27, C29 (see their interpretations in Tab.3), in particular, OS Linux is installed in it.

Step 1, Hacker 1. At the starting point, hacker 1 knows *IP* addresses of active hosts presented in data structure associated with variable x_2. He selects one of them and, while using *SKB1*, infers admissible activities at current step using iterations (7):

$$\begin{bmatrix} X^{(1)}{}_{2} \\ X^{(1)}{}_{14} \\ X^{(1)}{}_{15} \\ X^{(1)}{}_{21} \end{bmatrix} = \begin{bmatrix} \varnothing & \varnothing & \varnothing & \varnothing \\ [SS37]+[SS45] & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & [SS63]X_2X_{14} & \varnothing \end{bmatrix} \begin{bmatrix} x_2 \\ \varnothing \\ \varnothing \\ \varnothing \end{bmatrix} + \begin{bmatrix} x_2 \\ \varnothing \\ \varnothing \\ \varnothing \end{bmatrix} = \begin{bmatrix} x_2 \\ [SS37]x_2+[SS45]x_2 \\ \varnothing \\ \varnothing \end{bmatrix} = \begin{bmatrix} x_2 \\ x_{14} \\ \varnothing \\ \varnothing \end{bmatrix},$$

thus receiving the prompt that he can achieve the goal x_{14} in two ways, i.e. by use either activity (attack) *SS37* or *SS45*. These attacks have common preconditions C5 (*Network (host) firewall passes through tcp/ip-packets*) and *C9* ("*Connection "null sessions*") and different ones concerning with the type of OS, namely *C24* (OS *Windows*)–for attack *SS37*, and–*C25* (OS *Unix* or *Linux*)– for attack *SS45*. However, hacker 1 knows nothing about OS installed and tries first to execute the attack *SS37*. It fails because of OS installed in the host is not *Windows* and that is why he explores the second variant inferred in *SKB*, i.e. attack *SS45*. This attack is successful and hacker achieves the goal x_{14} – access to <*Shared Resources*> of the host.

Step 1, Hacker 2: The inference step looks as follows:

$$\begin{bmatrix} X_2 \\ X_{15} \\ X_{16} \\ X_{25} \\ X_{32} \end{bmatrix} = \begin{bmatrix} \varnothing & \varnothing & \varnothing & \varnothing & \varnothing \\ [SS52] & \varnothing & [SS50] & \varnothing & \varnothing \\ [SS49] & \varnothing & \varnothing & \varnothing & \varnothing \\ \varnothing & [SS75]X_2 & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & [SS84]X_2X_{15} & \varnothing \end{bmatrix} \begin{bmatrix} \varnothing \\ \varnothing \\ \varnothing \\ \varnothing \\ \varnothing \end{bmatrix} + \begin{bmatrix} \varnothing \\ \varnothing \\ \varnothing \\ \varnothing \\ \varnothing \end{bmatrix} = \begin{bmatrix} \varnothing \\ \varnothing \\ \varnothing \\ \varnothing \\ \varnothing \end{bmatrix}$$

This step results in no admissible activity and system can prompt to hacker (We do not here discuss how this can be done) that to succeed at the first step he needs to know the *IP* address of active hosts of the network, x_2. Through shared ontology hacker 2 is informed that hacker 1 knows x_2 and hacker 2 has to coordinate activity with the former. (In general case this means that the coalition has to coordinate its activity with other ones.) Hacker 2 asks and receives needed *IP* address selected by hacker 1 as the subject of attack. This coordination step is depicted in Fig.3 by dashed block arrow outgoing from the goal x_2 achieved by hacker 1.

Let us attract attention to the following aspect of the inference mechanism: (1) If the state of a hacker (a coalition) is not updated after an iteration of reasoning (no new achievable goals are inferred) then hacker (coalition) has to coordinate its activity with other ones; (2) All activities can fail and in this case it is necessary either to use coordination or postpone it while waiting help (coordination actions) from other hacker (coalitions) or changing of the environment state. An alternative is to use other *SKB*. An example of admissible course of action can be given based on *SKB* of hacker 2. For instance, after achievement of the goal x_{14} resulting from the first step, he cannot go forth if the goal x_{15} is not achieved by hacker 2 and needs either to wait this event or refuse of attack continuation to achieve the goal x_{21}.

Assume hacker 2 receives information x_2 and undertakes the new attempt:

$$\begin{bmatrix} X_2 \\ X_{15} \\ X_{16} \\ X_{25} \\ X_{32} \end{bmatrix} = \begin{bmatrix} \varnothing & \varnothing & \varnothing & \varnothing & \varnothing \\ [SS52] & \varnothing & [SS50] & \varnothing & \varnothing \\ [SS49] & \varnothing & \varnothing & \varnothing & \varnothing \\ \varnothing & [SS75]X_2 & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & [SS84]X_2X_{15} & \varnothing \end{bmatrix} \begin{bmatrix} x_2 \\ \varnothing \\ \varnothing \\ \varnothing \\ \varnothing \end{bmatrix} + \begin{bmatrix} x_2 \\ \varnothing \\ \varnothing \\ \varnothing \\ \varnothing \end{bmatrix} = \begin{bmatrix} x_2 \\ [SS52]x_2 \\ [SS49]x_2 \\ \varnothing \\ \varnothing \end{bmatrix}$$

The inference result is that hacker 2 can to undertake attacks [*SS52*] and [*SS49*] which, if successful, potentially allow to achieve the goals x_{15} (<*Users' Identifiers*> *are known*>) and x_{16} (<*Security identifier* (*SID*) *is known*>) respectively. For success of the first of attacks, [*SS52*], the precondition C5 (<*Network (host) firewall passes through tcp/ip-packets*>), C25 (<OS *Unix* or *Linux*>) и C29 (<*Service finger (port 79) is active and accessible*>) must be met. The [*SS49*] attack can be potentially successful if the same precondition C5 and also C8 (<*Windows* (*NetBIOS*)>) and C9 (<*Connection "null sessions"*>) are met. If hacker started from [*SS49*] attack he would fail because precondition C8 is not met (in the host OS *Linux* is installed. On the contrary, [*SS52*] attack can be successful because it is directed to the case of OS *Linux*. Let the hacker 2 (for instance, after the second attempt) achieves the goal presented by variable x_{15}.

Step 2, hacker 1. Before each new step, coalition has to transform its *SKB* to take into account the changes of the coalition state. Respectively, before the step 2, hacker has to transform *SKB1* to new state. In general case this is done as follows:

1. Failed activities of the previous step and their arguments are substituted in *SKB* matrix by symbol Ø–they cannot be used further in the current scenario.
2. The successful activities are substituted by Ø–there is no need to use them again. (This is done for convenience only, otherwise the chins will be duplicated.)
3. Nonterminals corresponding to the goals achieved are substituted by the respective terminal symbols.

While doing the above with *SKB1* of the hacker 1, the current state of it is transformed to the followings:

$$\begin{bmatrix} X^{(2)}{}_2 \\ X^{(2)}{}_{14} \\ X^{(2)}{}_{15} \\ X^{(2)}{}_{21} \end{bmatrix} = \begin{bmatrix} \varnothing & \varnothing & \varnothing & \varnothing \\ 1 & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & [SS63]x_2x_{14} & \varnothing \end{bmatrix}, \ \boldsymbol{X}^{(1)}(1) = \begin{bmatrix} x_2 \\ x_{14} \\ \varnothing \\ \varnothing \end{bmatrix},$$

Iteration with the *SKB1* matrix prompts to the hacker 1 that he needs coordination of further activity with the hacker 2 while waiting when the latter achieves the goal x_{15}. While receiving (if any) x_{15} (<*Users' Identifiers*>) hacker 1 infers via iteration that the goal x_{21} can be achieved if only the activity [*SS63*] is successful:

$$\begin{bmatrix} X^{(2)}_{2} \\ X^{(2)}_{14} \\ X^{(2)}_{15} \\ X^{(2)}_{21} \end{bmatrix} = \begin{bmatrix} \varnothing & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & [SS63]x_2x_{14} & \varnothing \end{bmatrix} \begin{bmatrix} x_2 \\ x_{14} \\ x_{15} \\ \varnothing \end{bmatrix} + \begin{bmatrix} x_2 \\ x_{14} \\ x_{15} \\ \varnothing \end{bmatrix} = \begin{bmatrix} x_2 \\ x_{14} \\ x_{15} \\ [SS63]x_2x_{14} \end{bmatrix},$$

After the attempt to realize this attack, hacker 1 makes sure that reaction of the environment (host security means) is negative (the precondition C24 is not held) and the ultimate goal x_{21} cannot be achieved it to use the skill represented in *SKB1*.

Step 2, hacker 2: New state of hacker 2 *SKB* after attacks of the step 1 is as follows:

$$\begin{bmatrix} X_2^{(2)} \\ X_{15}^{(2)} \\ X_{16}^{2} \\ X_{25}^{(2)} \\ X_{32}^{(2)} \end{bmatrix} = \begin{bmatrix} \varnothing & \varnothing & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & [SS50] & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & \varnothing & \varnothing \\ \varnothing & [SS75]x_2 & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & [SS84]x_2x_{15} & \varnothing \end{bmatrix}, \ \boldsymbol{X}^{(1)}(2) = \begin{bmatrix} x_2 \\ x_{15} \\ \varnothing \\ \varnothing \\ \varnothing \end{bmatrix}$$

The result of the next iteration with *SKB2* is as follows:

$$\begin{bmatrix} X_2^{(2)} \\ X_{15}^{(2)} \\ X_{16}^{2} \\ X_{25}^{(2)} \\ X_{32}^{(2)} \end{bmatrix} = \begin{bmatrix} \varnothing & \varnothing & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & [SS50] & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & \varnothing & \varnothing \\ \varnothing & [SS75]x_2 & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & [SS84]x_2x_{15} & \varnothing \end{bmatrix} \begin{bmatrix} x_2 \\ x_{15} \\ \varnothing \\ \varnothing \\ \varnothing \end{bmatrix} + \begin{bmatrix} x_2 \\ x_{15} \\ \varnothing \\ \varnothing \\ \varnothing \end{bmatrix} = \begin{bmatrix} x_2 \\ x_{15} \\ \varnothing \\ [SS75]x_2x_{15} \\ \varnothing \end{bmatrix} = \begin{bmatrix} x_2 \\ x_{15} \\ \varnothing \\ x_{25} \\ \varnothing \end{bmatrix}$$

He executes the inferred attack, [*SS75*], that is found out successful because all the preconditions are met.

Step 3, hacker 2: New state of *SKB2* after execution of attack of step 2 is as follows:

$$\begin{bmatrix} X_2^{(3)} \\ X_{15}^{(3)} \\ X_{16}^{(3)} \\ X_{25}^{(3)} \\ X_{32}^{(3)} \end{bmatrix} = \begin{bmatrix} \varnothing & \varnothing & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & [SS50] & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & [SS84]x_2x_{15} & \varnothing \end{bmatrix}, \ \boldsymbol{X}^{(2)}(2) = \begin{bmatrix} x_2 \\ x_{15} \\ \varnothing \\ x_{25} \\ \varnothing \end{bmatrix}$$

Next iteration looks as follows:

$$\begin{bmatrix} X_2^{(3)} \\ X_{15}^{(3)} \\ X_{16}^{(3)} \\ X_{25}^{(3)} \\ X_{32}^{(3)} \end{bmatrix} = \begin{bmatrix} \varnothing & \varnothing & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & [SS50] & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & [SS84]x_2x_{15} & \varnothing \end{bmatrix} \begin{bmatrix} x_2 \\ x_{15} \\ \varnothing \\ x_{25} \\ \varnothing \end{bmatrix} + \begin{bmatrix} x_2 \\ x_{15} \\ \varnothing \\ x_{25} \\ \varnothing \end{bmatrix} = \begin{bmatrix} x_2 \\ x_{15} \\ \varnothing \\ x_{25} \\ [SS84]x_2x_{15}x_{25} \end{bmatrix}$$

Hacker 2 is capable to realize the [*SS84*] attack, which is successful because all the preconditions are held. This means that hacker 2 achieved his ultimate goal of the coordinated multi-step attack that is the goal x_{32} –*<Admin (root) Password> is known>*, i.e. he got the access to the password of the host administrator.

Let us discuss the reasoning mechanism described by example and some conclusions. Let us assume that all the preconditions of the coalitions *SKB*s are held. For this case it is proved in [2], that (while slightly rephrasing and generalizing the results of [2]) after no more than *n* iterations (*n* is the dimensionality of the vector ***X***), the process is stopped due to one of the following reasons:

- Either a part or all ultimate goals are achieved. In this case all the positions of vector $\boldsymbol{X}^{(k)}$, $k \leq n$, corresponding to ultimate goals are assigned as terminals. The chains of intermediate activities with their arguments constitute the dynamically inferred scenario (represented in sequential–parallel form) of the ultimate goals achievement. Some position of the vector $\boldsymbol{X}^{(k)}$ can remain to be assigned symbol Ø, because not all goals appearing in *SKB* can be achieved.
- It is established that all or some goals are unachievable (coalition operation cannot achieve some or all ultimate goals) from the initial state of coalitions, e.g. in the above described case the goal x_{21} is unachievable.

The above statements are also held for the *SKB* with preconditions. Indeed, preconditions represent reactions of the environment. If to delete from *SKB* the nodes corresponding to activities, whose preconditions are not held for the task at hand, together with respective edges then the dimensionality of vector ***X*** specifying the resulting *SKB* cannot increase and therefore the complexity cannot also increase.

6. Conclusion

The paper objective is to present a new formal framework for specification of scenario knowledge bases supporting coalition operation dynamic planning, execution and coordination. It is based on formal grammar framework represented is a special, matrix, form enriched by attribute component and use of substitution operation allowing hierarchical specification of the scenario world. Although proposed framework is based on the ideas presented in the papers [1, 2], it contains a number of novelties oriented to their use for representation of scenario knowledge, dynamic inference of situational scenarios and coordination of activities of coalitions executing joint operation. It can be also used in multi-agent area for coordination of weakly coupled autonomous agents operating in a hostile or/and unpredictable environment.

The peculiarities of the framework are that it (1) is naturally integrated with the efficient inference mechanism that can be implemented in both *"data driven forward chaining reasoning"* and *"goal driven backward chaining reasoning"* modes; (2) allows to dynamically infer scenarios of coordinated activities of coalitions executing joint operation; (3) takes into account a hostile and unpredictable nature of environment reactions on activities of coalitions; (4) allows hierarchical specification of scenarios; (5) allows to determine the coordination points of coalitions.

The proposed formal framework can be simply extended to the cases when the environment reaction is of probabilistic or fuzzy nature and used for coordination and control of coalition operations as well as framework for simulation purposes.

We are now at the beginning of implementation and thus future works will be manifold. The first task is to in-depth development of algorithms aiming at combined use of both *"data driven forward chaining reasoning"* and *"goal driven backward chaining reasoning".* The second one is development of a software tool destined for design and implementation of scenario knowledge bases and efficient inference mechanisms. The third task is exploration of the framework properties and its validation on the basis of different applications of both coalition operation and multi-agent system research areas.

References

[1] N. Chomsky and M.Schutzenberge. *Algebraic Theory of Context Free Languages. Chapter in Cybernetical Collection of Articles*, New Series, 1966, vol.3, 118-161 (Russian Editions)

[2] M.P.Georgeff and A.S.Rao. BDI Agents: From Theory to Practice. *In Proceedings of the First International Conference on Multi-Agent Systems* (ed. V. Lesser). AAAI Press/The MIT Press, 1995, 312-319

[3] V.Glushkov, A.Zeitlin, E.Yustchenko. Algebra, Languages, Programming. Naukova Dumka, Kiyev, 1975 (in Russian)

[4] V.Gorodetsky and A.Tarakanov. Computations Planning Based on Inference in Attribute Context Free Grammars. *In A.Slissenko (Ed.) Mathematical Methods of Development and Analysis of Algorithms*. Leningrad, Nauka, 1990, 37-48 (in Russian).

[5] V.Gorodetski and I.Kotenko. Attacks against Computer Network: Formal Grammar-based Framework and Simulation Tool. *In Proceedings of the 5 International Conference "Recent Advances in Intrusion Detection", Lecture Notes in Computer Science*, vol. 2516, Springer Verlag, pp.219-238, 2002.

[6] W. van der Hoek. Logical Foundation of Agent-based Computing. Multi-agent Systems and Applications. *EASSS 2001, Lecture Notes in Artificial Intelligence,* vol.2086, 2001, 50-73

[7] D.Rosenkranz. Matrix Equationsand Normal Form of Context Free Grammars. *Journal of ACM*, 1967, vol.14, 501-507.

[8] A Tarakanov. Matrix Method for Automatic Synthesis of Programs. *Izvesija VUZ, Priborostroeniye*, vol.31, #10, 1988, 21-25 (in Russian).

[9] M.Wooldridge. Reasoning about Rational Agents. MIT Press, Cambridge, MA, 2000.