Mathematical Problems of Computer Science 31, 169--177, 2008.

Anomalies Dynamic Analysis and Correction Software*

Edward Pogossian[1,2] and Arthur Grigoryan[1]

[1]Institute for Informatics and Automation Problems of NAN RA,
[2]State Engineering University of Armenia,
E-mail: epogossi@aua.am

Abstract

The research is aimed to develop an effective Anomalies Dynamic Analysis and Correction Software (ADACS) for Grid Armenia. A software is developed that in addition to predetermined and fixed forms of protection of variety of servers generates game trees of possible anomalies and elaborates recommendations to avoid them by analyzing possible strategies throughout the game trees and searching the best correction strategies. Experiments on correction of anomalies in overfilling the memory of the cluster of IPIA are processed.

1. Introduction and Overview

1.1. Grid computing is a technology addressed to the definition and implementation of a large scale resource sharing environment. Due to its collaborative and distributed nature, the grid environment requires a large support to guarantee security in its transactions. Security management in the grid environment is complicated by the need to establish security relationships between a large number of dynamically created subjects and across distinct administrative domains, each with its distinct resources and its own local security policy. Because each participant often has to inter-operate with existing local systems the grid security model has to be integrated with existing local systems, its security mechanisms have to inter-operate with different hosting environments and, particularly, establishing trust relationship among different domains [1 -5].

There are two main categories of intrusion detection methods: the ***detection of abuses*** and the ***detection of anomalies.***

The abuses detection systems contain the descriptions of attacks ("signatures") and search for the correspondence to these descriptions in the tested data stream, in order to find out the development of a known attack. The main advantage of such systems is in concentration on analysis of tested data and in producing few false alarms. The disadvantage is in definition of only known attacks, which have their proper signatures [6]

The anomalies detection systems (ADS) are based on the interpretation of abnormalities as potential attacks. The advantage of this approach is in revealing attacks unknown before as well as the anomalies in the work of the system by themselves.

The disadvantages are in a large amount of false alarms, in decisions making mechanisms about attacks and in its monolithic architecture, which hard adapts to dynamic changes in the system configuration [7].

Currently the development of ADS is facing a principal difficulty achieving a consistency and compatibility in the way ADS and humans use and exchange knowledge.

1.2. We aim to progress towards developing human knowledge acquisition by competitive ADS that elaborate strategies.

We develop the approach announced in [11] and study the problem solving for the subclass of optimal strategy provision problems where the *space* of hypothesis of *solutions* can be specified by *game trees* (SSGT).

* A-1451 Grid Armenia ISTC Project for the Institute for Informatics and Automation Problems of NAN RA

Viability of the approach was demonstrated for the intrusion protection SSGT problems and the *Intermediate Goals At First* (IGAF) [8] algorithms.

In [11] we have defined Personalized Planning and Integrated Testing (PPIT) algorithms able to elaborate actions in target situations dependent on a variety of categories of expert knowledge. We experiment with Reti and Nodareishvili etudes requiring by Botvinnik, intensive expert knowledge based analysis not available to conventional chess programs. We find that PPIT1 programs, in principle, are able to acquire the contents (meanings) of units of vocabulary used by experts and allow one to tune the contents properly to acquire the solutions of those etudes and play them properly.

1.3. This research is aimed to develop an effective Anomalies Dynamic Analysis and Correction Software (ADACS) for the Grid Armenia that in addition to predetermined and fixed forms of protection of variety of servers generates game trees of possible anomalies and elaborates recommendations to avoid them by analyzing possible strategies throughout the game trees and searching the best correction strategies.

As the result of the research effective and efficient anomalies detection and protection software will be developed which includes the following basic tasks:

- specification of classes of typical anomalies,
- developing programs elaborating optimal protection strategies for classes of anomalies,
- developing administrator's supervision and control programs,
- implementation of ADACS in the Grid Armenia.

In the paper we present current state of ADACS development including description of the following components:

- classes of some cluster anomalies,
- grid version of PPIT program for elaborating optimal protection strategies for anomalies,
- administrator's interface to supervise PPIT grid,
- experiments on memory overfilling type anomalies correction in the IPIA cluster by using PPIT grid.

2. Classes of Anomalies

2.1. The analysis of IPIA cluster administrator's work revealed, particularly, the following types of anomalies needed for corrections:

- overfilled memory and buffers,
- deadlock anomalies.

2.2. ***Overfilled memory and buffers.*** In the operating time of programs arise situations with memory overflow that for different reasons don't determine necessary corrections. The best way to resolve these situations is to pass a message to the manager of the resources.

To operate with these types of anomalies where the critical resource is the memory we will use the following designations $L_1(R_1)$

Resource R_1: MemoryBuffer - the memory buffer
MemoryBufferSize - the size of the buffer of memory
LoadedMemoryBufferSize - the filled size of the buffer of memory.

Qualifier $L_1(R_1)$ for the states of resource R_1:

$$L_1(R_1) = 1 - (\text{MemoryBufferSize} - \text{LoadedMemoryBufferSize}) / \text{MemoryBufferSize}$$

Qualifier $L_1(R_1)$ maps a state of the buffer of memory on a segment [0, 1] by a principle increase of an amount of the occupied memory. For example, $L_1(R_1) = 0$ means that all memory is in a free state and $L_1(R_1) = 1$ means then all memory is occupied.

2.3. ***The deadlock anomalies*** are widely studied but have no acceptable solution, yet. The difficulties are, particularly, the following.

The message passing interface (MPI) is a commonly used application programming interface for the development of portable parallel programs. It is easy, however, to create MPI programs that are prone to deadlock. It is desirable to be able to detect these deadlocks in running programs. It is further desirable

to perform this deadlock detection in a distributed manner, without assuming the existence of shared memory for communication.

Below is an example showing a communication pattern that leads to a deadlock.

Process 0	Process 1
Recv(1)	Recv(0)
Send(1)	Send(0)

In this example, the RECV call is blocked. Therefore each process is waiting to receive from the other and blocks there waiting for a corresponding send. The send can never be executed, and both processes are blocked at the RECV resulting in a deadlock.

Another example that is unsafe to use is:

Process 0	Process 1
Send(1)	Send(0)
Recv(1)	Recv(0)

This situation is correct logically, but it depends on how the blocking is implemented for the SEND and RECV. If SEND blocks until the data are copied out of the sender buffer, then a deadlock can happen if the system doesn't have enough buffer to allocate both sends at the same time. If the system can allocate enough buffer for both sends to go, then they will return, and the RECV will start. The order depends on the system resources and therefore is unsafe to use.

The following scenario is always safe because each SEND is matched by a corresponding RECV before the other SEND starts.

Process 0	Process 1
Send(1)	Recv(0)
Recv(1)	Send(0)

Our idea to advance in the Deadlock problem is based on its reduction to the combinatorial games based model and application of the effective strategy elaboration methods.

On this stage our efforts are directed on creating PPIT based solutions for Deadlock anomalies.

3. Game Tree Model for Grid Anomalies Analysis

3.1. We define Grid anomalies correction problem in frame of the SSGT class [11] with the following requirements:

There are (a) interacting actors (players, competitors, etc.) performing (b) identified types of actions in the (c) specified moments of time and (d) specified types of situations.

There are identified benefits for each of the actors, descriptions of situations legal for the actors' actions and those that are transformed after actions.

For such problems with given arbitrary situation x and actor A, who is going to act in x, we can generate corresponding game tree GT(x, A) comprising all games started from x.

The games represent all possible sequences of legal actions of the players and situations that they can create from given initial, or root situation x. In our consideration the games are finite and end by one of goal situations of the problem.

Assuming A plays according to a deterministic program, a strategy, the GT(x, A) represents, in fact, all possible performance trees of the strategies from the x. In that sense the GT(x, A) determines the space of all possible solutions starting from the x situation.

Given criterion K to evaluate the quality of strategies we can define the best strategy S*(x, A) and corresponding best action of A from x.

3.2. The above requirements to SSGT problems we interpret for Grid environment as the following:

The game tree model for grid anomalies dynamic analysis is constructed by the same principles as developed in [8, 9] for intrusion protection problems. It considers a game between playing in turn two sides with opposite interests - the attacker (A) and the defender (D), described by a set of states and a collection of conversion procedures from one position to another defined as the following.

System resources are particularly the processor time, the size of TCP buffer, and a number of incoming packages. Let R = {r} be a non-empty set of the system resources and Q is a set of resource

parameters. Different measuring scales, such as seconds, bytes, numbers of incorrect logins or incoming packages, etc., are used to measure different parameters. Each $r \in R$ is associated with a pair $< q; w >$, a real system resource, where $q \in Q$, $w \in W$ and W is a set of possible scales.

A criterion function is an arbitrary function f with the range of values $Z=[0,1]$ and F is the set of such functions f.

A local system resource state on a non-empty set $R' \in R$ is called the value $e \in Z$ of the criterion function $f \in F$ on this set: $e = f(r_1, r_2, ..., r_k)$, where $R' = (r_1, r_2, ..., r_k) \& \varnothing \neq R \subseteq R$.

The *local state* is called *normal* if $e = 0$ and *critical* if $e = 1$; L denotes the set of local states e. Intuitively, the criterion functions measure the "distance" between current states and those that are considered as normal. A system state on a non-empty set $L' \in L$ is called the value $s \in Z$ of the criterion function $g \in F$ on this set: $s = (e_1, e_2, ..., e_n)$, where $L' = (e_1, e_2, ..., e_k) \& \varnothing \neq L' \subseteq L$. The *state* is called *normal* if $s = 0$ and *critical* if $s = 1$; S denotes the set of states s.

The main goals of the attackers and defenders are to bring the system in the critical states and avoid them, correspondingly.

Let us call an arbitrary function $p(s_i, s_j)$, the ranges of definition and values of which are subsets of R, *a conversion procedure* of system from the state s_i to s_j:

$p(s_i, s_j) : \{\{r_1, r_2, ..., r_k\} \rightarrow \{r'_1, r'_2, ..., r'_k\}\}$, where $\{r_1, r_2, ..., r_k\} \in R \& \{r'_1, r'_2, ..., r'_k\} \in R$.

Let P be the set of conversion procedures, P_a and P_d be its subsets for the attacking and the counteracting sides, $p_a(s_i, s_j) \in P_a$ and $p_d(s_i, s_j) \in P_d$ are the conversion procedures from the state $s_i \in S$ to $s_j \in S$ for the attacking and the counteracting sides, correspondingly.

The counteraction game model is represented by "AND/OR" tree G(S,P), where S and P are finite, non-empty sets of all states (nodes, vertices) and all possible conversion procedures (edges, ribs), correspondingly (Fig. 1).

At first the attacker moves from the initial state S_0 then the defender replies in turn. Thus, the initial node S_0 is an "AND" type. The terminal nodes correspond to the winning states of the defender.

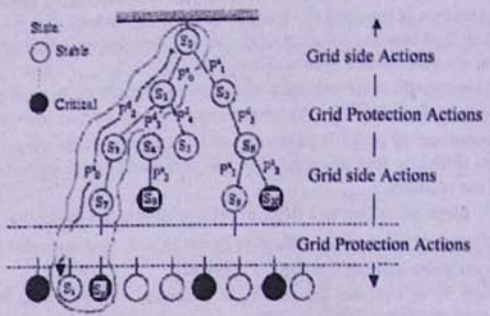

Figure 1. A game tree model for the Grid anomalies analysis and correction.

3.3. In the Grid environment the players of the above model are interpreted as the one which can cause some anomalies (A) and unify the users that are processing some tasks in the Grid at the moment, technical disturbances, incorrect loading in the Grid, etc., and the one representing the administrator's functions aimed to correct the anomalies.

Despite the basic goal of the users included into a player's side is the solution of tasks, A is considered in the anomalies causing projection, i.e. as having the goal to create anomalies. Apparently, the goal of D is to avoid or correct the anomalies.

The actions of A player include, particularly, the following ones: starting new processes, ending some ongoing processes, sending or receiving messages by message queue, prolongations or completion of performance of the programs.

The actions of D player include, particularly, the following ones: termination of some processes, termination of performance of some programs, prolongation of ongoing operations, exposition of critical situations or passages, exposition of controlled resources.

4. Developing PPIT Grid

4.1 The PPIT programs aimed to acquire strategy expert knowledge to become comparable with a human in solving hard problems [10]. In fact, the following two tasks of knowledge acquisition can be identified in the process:

- construction of shells of the programs allowing to acquire the contents of units of chess vocabulary and,
- construction of procedures for regular acquisition of the contents of the units by the shells.

We require an effective shell to possess the following characteristics:

- be able to store typical categories of common chess knowledge as well as the personalized one and depend on them in strategy formation
- be able to test approximate knowledge based hypothesis on strategies in questioned positions by reliable means, for example, using game tree search techniques.

The second task we plan to solve in the following two stages:

- to prove that the shells, in principle, can acquire the contents of units of vocabulary used by chess players and allow to tune them properly to solve problems with intensive chess expert knowledge
- to develop procedures for regular acquisition of the contents of those units .

4.2. We design shells of PPIT1 programs as a composition of the following basic units [12]:

- Reducing Hopeless Plans (RHP)
- Choosing Plans with Max Utility (CPMU)
- Generating Moves by a Plan (GMP)

Given a questioned position P_1 and a store of plans, RHP recommends to CPMU a list L_1 of plans promising by some not necessary proved reasons to be analyzed in P_1. The core of the unit is knowledge in classification of chess positions allowing identify the niche in the store of knowledge the most relevant for analysis of the position. If the store of knowledge is rich and P1 is identified properly it can provide a ready-to-use portion of knowledge to direct further game playing process by GMP unit. Otherwise, RHP, realizing a reduced version of CPMU, identifies L_1 and passes the control to CPMU.

CPMU recommends to GMP to continue to play by current plan if L_1 coincides with list L_0 of plans formed in the previous position P_0 and changes in P_1 are not essential enough to influence on the utility of current plan.

If changes in P_1 are essential, CPMU analyzes L_1 completely to find a plan with max utility and to address it to GMP as a new current plan. Otherwise, CPMU forms new complementary list $L_1 / L_1 * L_0$ from the plans of L_1 not analyzed, yet, in L_0, finds a plan with the best utility in that list and comparing it with utility of current plan recommends the one of them with a higher utility.

To calculate utilities of the attribute, goal and plan type units of chess knowledge, we represent them as operators over corresponding arguments as follows:

- for basic attributes the arguments are characteristics of the states of squares in the questioned positions, including data on captures of pieces, threats, occupations, etc.
- for composed attributes, including concepts and goals, the arguments are subsets of values of basic attributes relevant to the analyzed positions

- for plans the arguments are utilities of the goals associated with realization of those plans.

Utilities of arguments of basic attributes are calculated by trajectory-zones based technique (TZT) [12] originally suggested for estimating utilities of only captures of the opponent pieces in chess. For example, to chose capture with max utility TZT at first chains the moves to each piece of the opponent (trajectories) without accounting possible handicaps for real capturing. Then, using all available knowledge plays the zones of the game tree "induced" by the trajectories followed by estimation of the values of the zones to choose the best.

The utility of units of knowledge the operators assemble from utilities of corresponding arguments in some predetermined ordering. Thus, each operator can provide by a request the arguments which are analyzed at the moment.

For example, realizing current plan the shell can determine the goal in the agenda which in turn determines basic attributes to be considered followed by indication of the arguments of those attributes.

Utility estimation operators rely on the principle of integration of all diversity of units of knowledge the shell possess at the moment. In fact, the operators represent a kind of expert knowledge with a variety of mechanisms and leverages to make them better. Along with dynamically changed parametric values of pieces they can include rules, positions with known values and strategies to realize them, other combinatorial structures. To estimate expected utilities the operators take into account the cost of resources necessary to get them.

In [9] the units of knowledge are realized as OO classes with specialized interfaces for each type of knowledge and one common for the shell itself.

5. Experimenting with PPIT Grid

5.1. The game tree of dynamic analysis of current grid situation, detection of its anomalies and elaboration of strategies for their correction are based on the data of the Info Center.

The data are available by specialized server which in a centralized way monitors resources of virtual organization relevant to Grid, or its constituents.

To generate a game tree relevant to the most suspicious at present resources of the Grid ADACS like the software in [8, 9] includes single-level and multilevel identifier-classifiers of the states of the constituents of the base system.

The classifiers are distributed by the constituents of the Grid and "fuse"data from them to determine the list of resources suspicious to be affected. Then, the lists of offensive and defensive actions is generated followed by generation of corresponding game tree, searching the best strategy and doing the decision recommended by that strategy.

We rely on the following concepts of "trajectory of an attack" and "zone of counteraction".

The trajectory of an attack is a subtree $G_a(S^*, P^*)$ (Fig.2), where S^* is a subset of the system states $S^* \in S$ and P^* is a subset of the actions, consisted of an offender's conversion procedures P_a and a defender's normal conversion procedures P_{dn}, i.e. $P^* \in P_a \cup P_{dn}$, $P^* \subseteq P$, $P^* \neq \varnothing$.

The zone of counteraction is a subtree $G_z(S^*, P^*)$ built around the graph of the trajectory of an attack $G_z(S^*, P^*)$, i.e. $G_a \in G_z$, where S^* is a subset of the system states $S^* \in S$, which belong to the trajectory of an attack, hence $S^* \in S^*$, and P^* is a subset of the actions, which consist of the conversion procedures, defined on the trajectory of an attack, P^* and the defender's special conversion procedures P_{dz}, i.e. $P^* = P^* \cup P_{dz}$, $P^* \subseteq P$, $P^* \neq \varnothing$, hence $P^* \subseteq P^*$.

5.2. On this step of development we try to assure the viability of PPIT Grid using "goal" and "rule" types of expert knowledge similar to [9] described as the following:

The goals:

1. the *critical vs. normal states* are determined by a range of values of the states of the system; for example, any state of the system with a value of corresponding criterion function, that is more or equal to some threshold, may be determined as a critical goal,

2. the *suspicious vs. normal resources* are determined by a range of states of the classificatory of the resources; combinations of values of the classificatory identified as suspicious or normal induce signals for appropriate actions.

The rules:

1. Identify the suspicious resources by the classifiers and narrow the search to corresponding game subtree.
2. Avoid critical states and tend to the normal ones.
3. Normalize the state of the system. First, try such actions of the defender that influence on the resources caused current change of its state and if they don't help try other ones.

3 In building game subtree for suspicious resources use,

- defending actions able to influence on such resources,
- use normal actions until there is no critical states,
- if some defensive actions were used on previous steps decrease their usage priority,
- balance the parameters of resources by keeping them in the given ranges of permitted changes.

4. Let us present an example of overcoming memory overflow type anomalies by the ADACS.
ne A and D sides operations are the following:

A: fill one cell of memory, fill two cells of memory, fill random cells of memory

D: allow capturing memory, deny capturing memory, increase memory buffer size, release emory randomly.

We use standard min max technique with alpha-beta pruning based on the range of itical/normal state values introduced as the goal 1. Current node is created and the value of its local state lculated. If the node is terminal, the local state value is compared with sibling nodes, and their max ither min) value is sent to the parent node.

By the command: Find the most suspicious resource, the program indicates the memory.
ne consequent steps are the following:

build the game subtree for the memory resource starting from the root state of the tree and using the 4th group of rules determine the trajectories of attacks (Fig.2).

calculate the values of the terminal states of the tree, find the values of others by minmax procedure and determine the best minmax action from the root state .

determine the trajectories of attacks induced by the best action from the root of the tree to its critical states and consider them as targets .

build the zones of counteractions for the target trajectories using the 4th group of rules and rule 5th, then calculate the values of the states of the corresponding subtree using the minmax (Fig.3).

choose the defender's action from the root as the one leading to the state with min value, i.e. to the most stable state estimated by the minmax.

end the defense analysis and wait for the attacker's actions.

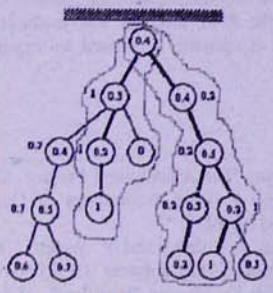

Figure 2. Actions trajectory

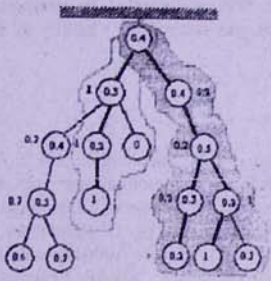

Figure 3. Defense zone

Fig.4 presents final interface of the process of overcoming memory overflow anomaly where at the beginning the memory is supposed empty and the A, Grid side, starts the actions.

The estimates of the situation caused by the A actions followed by actions of the defending administrator's, D side are calculated in accord with the above algorithm.

Yellow frames contour actions of the sides, S_A_K, S_D_K are auxiliary symbols to tune the system for experiments, D_K and A_K are situations with A and D actions, correspondingly.

Final data of the experiment at the bottom of the figure are the following:

Action = Tree Generation

Tree Depth = 6, Steps Count = 3, Searched Nodes Count = 12267, Critical Value of the state of the system = 0.7

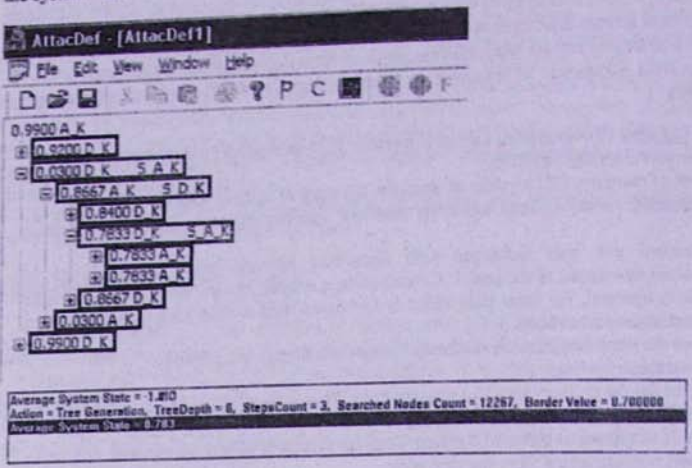

Fig.4. An example of the interface of memory overflow and correction.

Conclusion

The research is aimed to develop an effective anomalies dynamic analysis and correction software – ADACS, for Grid Armenia. Two classes of typical anomalies: overfilling the memory and Deadlock, focusing IPIA cluster of the Grid Armenia are considered. Grid anomalies correction problem in frame of the SSGT class is defined followed by interpretation of the PPIT ideology and methods for Grid anomalies detection and correction. Viability of PPIT grid is currently examined by experiments in anomalies detection.

References

1. H. V. Astsatryan, Yu. H. Shoukourian and V. G. Sahakyan, "The ArmCluster1 Project: Creation of high performance computation cluster and databases in Armenia Proceedings of Conference", *Computer Science and Information Technologies*, pp. 376-379, 2001.
2. R. Butler, D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer, and V. Welch, "Design and deployment of a National-Scale Authentication Infrastructure", *IEEEComputer, 33*(12):60-66. 2000.
3. G.A. Bolcer and G. Kaiser, "SWAP: Leveraging the Web To Manage Workflow", *IEEE Internet Computing*,:85-88. 1999.
4. I. Foster, C. Kesselman and S. Tuecke, "The Anatomy of the Grid Enabling Scalable Virtual Organizations", *Intl Journal, Supercomputer Applications* 2001.

I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke, "A security architecture for computational grids". In *ACM Conference on Computers and Security*, 83-91, 1998.
K. Ilgun, R.A. Kemmerer and P.A. Porras, "State transition analysis: A rule-based intrusion detection system", *IEEE Trans. Software* Eng. vol. 21, no. 3, Mar. 1995.
F. Martinelli, P. Mori and A. Vaccarelli, "Improving grid services security with fine grain policies", *Institute for Informatics e Telematics, Pisa, Italy*, 2003.
E. Pogossian, A. Javadyan and E. Ivanyan, "Effective discovery of intrusion protection strategies", *Lecture Notes in Computer Science, AIS-ADM-05: The International Workshop on Autonomous Intelligent Systems -Agents and Data Mining* St. Petersburg, Russia http://space.iias.spb.su/ais05/, LNAI 3505, pp.263-276, June 6-8, 2005.
E. Pogossian and A. Javadyan, "A game model for effective counteraction against computer attacks in intrusion detection systems", *NATO ASI 2003, Data Fusion for Situation Monitoring, Incident Detection, Alert and Response Management*, Tsahkadzor, Armenia, pp.30, August 19-30, 2003.
E. Pogossian, V. Vahradyan and A. Grigoryan, "On competing agents consistent with expert knowledge", *Lecture Notes in Computer Science, AIS-ADM-07: The International Workshop on Autonomous Intelligent Systems - Agents and Data Mining*, St. Petersburg, Russia, pp. 229-241, 2007.
E. Pogossian, "Combinatorial game models for security systems. NATO ARW on Security and Embedded System", Porto Rio, Patras, Greece, Aug. 8-18, 2005.
M. Botvinnik, *Computers in Chess: Solving Inexact Search Problems*. Springer Series in Symbolic Computation, with Appendixes, Springer-Verlag: NY, 1984.

Շեղումների դինամիկ վերլուծության և վերացման համակարգ

Է. Պողոսյան և Ա. Գրիգորյան

Ամփոփում

Grid միջավայրում շեղումներից պաշտպանության պրոբլեմի համար դիտարկվում են արցեր, կապված որոշումների ընդունման ծառերի դինամիկ վերլուծության հետ,: Ստեղծված է ռագիր, այդպիսի ծառերի դինամիկ գեներացիայի համար, և ընարագրված են ստացված սպերեմենտալ փորձարկման արդյունքները՝ հիշողության գերհագեցման պրոբլեմի օրինակի ամար: