Experiments Validating the Be-Have-Do Meaning Presentation Model and Matching Algorithm for Competing and Combating Problems

Karen Khachatryan
State Engineering University of Armenia
Yerevan, Armenia
e-mail: khkarens@gmail.com

Sedrak Grigoryan
Laboratory of Cognitive Algorithms and Models, IPIA, NAS RA,
Yerevan, Armenia
e-mail: addressforsd@gmail.com

Tadevos Baghdasaryan
Laboratory of Cognitive Algorithms and Models, IPIA, NAS RA,
Yerevan, Armenia
e-mail: tbs_@mail.ru

ABSTRACT
The paper reports experiments on viability of language driven be-have-do model to present and process expert knowledge used in solving problems of the class where Space of Solutions can be Represented as Reproducible Game Trees (SSRGT).
The model is based on the structure of be-have-do relationships between language categories in English and corresponding meanings revealed in realities as well as a novel model of the meaning presentation - Graph of Abstracts (GA).
We present experiments providing a positive evidence on viability of the model for the following SSRGT problems, namely, Chess, Management Support (TAC game) and Intrusion Protection problems.

Keywords
Meaning processing, knowledge, be-have-do model, experiment, management, chess, intrusion protection

1. INTRODUCTION
We study the knowledge-based strategies for the SSRGT class [1]. Unified SSRGT specification of problems makes possible to design a unified Solver for the problems of the class. Solver of the SSRGT problems is a package [2], [3], [9] aimed to acquire strategic expert knowledge to become comparable with a human in solving hard combinatorial competing and combating problems.
One of the important tasks of expert knowledge acquisition is the construction of procedures for regular acquisition of the meanings of units of vocabulary (UV) of problems by the package. The meanings are defined in [4] as certain assembles of regularities which have, at least, identifiers, names, and do combine with other meanings by *have, be, do* types of links, relationships, subjects to be varied in time, aspects, modality and certain other syntax categories. The problem of modeling and processing the meanings of lexical units is especially singled out also in the recent researches of the semantic/meaning-based search in the web.
Yet in 2009 it was reported [5] that the performance of web search engines was low for both keyword-based search engines and the semantic search engine. Though, the searching giants, like Google with its Knowledge Graph [6] and Microsoft with its Satory [7] in 2012 introduced the semantic/meaning-based, extensions to their search engines, the problem of understanding the natural language sentences as well as supporting languages *but* English still remain as research challenges in the semantic web search.
The UNL project [8] is another example of the efforts to enhance computers to understand what is meant by natural languages. In [9] we developed a novel model of the meaning presentation based on meaning processing and English syntax models in [4]. Unlike other works [10] which use a linguistic connection between words to evaluate the meanings of words, we connect the words with abstracts and use the language semantic connections to present the meanings of abstracts (and consequently, the meanings of words - connectors, associated with them).
This paper is devoted to the experiments validating the developed model of meaning presentation and processing.
Particularly, we detail the procedure of building new nodes and connections within the GA during meaning acquisition and analyze the growth of the graph for three different problem instances of SSRGT class, namely, chess, management and Intrusion Protection.
Next we discuss the iterative matching procedure of different types of abstracts and report relevant measurements.

2. PRESENTATION OF CHESS KNOWLEDGE
For chess experiments we consider the modeling of Rook Abilities concepts from [11]. Although it is a relatively simple concept in chess, it transparently demonstrates the presentation of a meaning in our be-have-do model. Its modeling leads to a construction of various types of nodes and relations as well as underlines the optimizations developed for the model.
Particularly, the definition of Rook Abilities requires the modeling of both specialized and generalized meanings. For example, how to model the meaning of "rook move" which would include all possible moves of a rook? - this is basically the part of Rook Abilities concept. The other difficulty is the definition of a "group" meaning. For example the definition of empty fields on the same line. This is simple by its specification, but it has a specific structure when representing its meaning as well as when matching it in situations.

2.1. Basic Chess Concepts
As for every chess concept, we need to have defined nucleus concepts, the basics of the game. As discussed in [9] the basics are chess board, figures and the sides (white, black players). We will skip some of the definition details due to the limits of the paper.
The board can be defined as a composition of fields, where each field has its coordinates. The figure has a color and a type, which identifies whether this is a pawn, a knight, or some other type of figure.
It is important to note that all these nucleus abstracts will formulate the entry nodes or roots of the graph of abstracts.
They basically play two important roles:

- they serve as connectors between the knowledge and the situations,
- they serve as nucleus units for construction of other abstracts.

Clarifying the meanings and the roles of nucleus abstracts as the next step we define the Field abstract by simple composition of these nucleus ones.

As the value ranges are not specialized for Field abstract, only *have* connections to the existing nucleus nodes are built. Therefore, the relations of Field can be presented as {*be*=""; *have*={fc:FigureColor, ft:FigureType, cordx:X, cordy:Y}; *do*=""}. We define Figure as a Field, which has specialized color and type, which leads to creation of two new nodes: Figre.fc (figure color) and Figure.ft (figure type). Figure abstract will have the following relations: {*be*="Field"; *have*= {fc:Figure.fc, ft:Figure.ft, cordx:X, cordy:Y}; *do*=""}.

2.2. Rook Abilities Concept

Other figures are defined like Figure abstract. For example Rook is derived from Figure, and specifies new value of Figure.ft attribute (=4). Other values are not changed. So its integration is similar to the integration of Figure abstract.

To define the more complex units of meanings, we define a set abstract called EmptyInterval. Variations of this abstract (the specializations) can represent different successive empty fields between the given positions. It has an element of Field type with EmptyInterval.element.ft=0 and minCount=1, maxCount=8. To create EmptyInterval first the algorithm creates EmptyInterval.element.ft abstract with a value = 0, and adds a *be* relation to FigureType, afterwards, it creates EmptyInterval.element abstract, adding a *be* relation to Field and *have* relations to X, Y, FigureColor and EmptyInterval.element.ft nodes. Once EmptyInterval's attribute is created the algorithm creates *have* connection from EmptyInterval to EmptyInterval.element. This abstract, in fact, can contain any combination of empty fields where number of Fields is 1 to 8. Therefore, it can be described by the following: EmptyInterval {*be:*""; *have*: {element:Field}; *do*: ""; minCount=1; maxCount=8}.

Continuing the definitions we get to a virtual abstract called FieldsBetween. The meaning of this abstract is to describe a Figure and a Field on either the same vertical or on the same horizontal, while in either case there shall be only empty fields (or no field) between them. Consequently, it is defined as a composition of a Figure and a Field abstract whose X and Y coordinates are undefined. As this is a virtual abstract, it doesn't create any *have* relation to the existing abstracts (it interacts with the rest of GA with its specifications and usages [1]). There are 4 specifications of FieldsBetween abstract which include EmptyInterval abstract and 4 others which do not (in the case of the latter, Field and Figure are neighbors). The difference between specifications stands within their coordinates: 1) Fig.X=Fld.X and Fig.Y>Fld.Y, 2) Fig.X=Fld.X and Fig.Y<Fld.Y, 3) Fig.X>Fld.X and Fig.Y=Fld.Y, 4) Fig.X<Fld.X and Fig.Y=Fld.Y. We will discuss only the first case, while the others can be specified similarly.

Fig 1. A specification of FieldsBetween abstract

We define FieldsBetween1 (Fig 1) in the following way. Obviously it is derived from FieldsBetween virtual abstract (Fig 2). Here Field has already specified values for X and Y according to the above description of this case, Field.X=Figure.X and Field.Y<Figure.Y. EmptyInterval also defines new dependencies here: EmptyInterval.element.X=Figure.X, EmptyInterval.element.Y=IN[Field.Y+1, Figure.Y-1] (this rule shows that Y attribute range is between Figure and Field). Moreover, we can add one more specification here, EmptyInterval.element.ft=[4,5], which indicates that the concept is applicable only for Rook and Queen. EmptyInterval.minCount=Figure.Y–Field.Y-1 and EmptyInterval.maxCount=Figure.Y–Field.Y-1, where the last two specify the number of fields between Figure and Field. During the integration into the GA the algorithm first creates *be* relation to the FieldsBetween virtual node, then *have* relation to Figure and Field nodes and EmptyInterval. The new dependencies between Figure's, Field's and EmptyInterval's attributes are kept in the FieldsBetween1 node. Consequently, the abstract will be represented by the following: FieldsBetween1 is {*be*: FieldsBetween; *have*: {fg:Figure, f:Field, ei:EmptyInterval}; *do*:""; f.X=fg.X; f.Y<fg.Y; ei.element.X=fg.X, ei.element.Y IN[f.Y+1, fg.Y-1], ei.minCount=fg.Y–f.Y-1, ei.maxCount=fg.Y–f.Y - 1}.

The differences of the other four specifications (which do not include EmptyInterval abstract) are as follows: 1) Fig.X=Fld.X and Fig.Y=Fld.Y+1, 2) Fig.X=Fld.X and Fig.Y=Fld.Y-1, 3) Fig.X=Fld.X+1 and Fig.Y=Fld.Y, 4) Fig.X=Fld.X-1 and Fig.Y=Fld.Y.

Thus, FieldsBetween6 will look like this: {*be*: FieldsBetween; *have*: {fg:Figure, f:Field}; *do*: ""; fg.X = f.X, fg.Y = f.Y - 1}.

After the above definitions the graph of abstracts will have the structure represented in Fig 2.

RookAbilities can be defined as an abstract having a single FieldsBetween attribute and specifying the type of FieldsBetween.Field.ft=4 (rook). It will have the following representation: {*be:*""; *have:*{fb:FieldsBetween}; *do:*""; fb.fd.ft=4}.

2.3. Presentation of Other Concepts

Similar to the RookAbilities we can define the BishopAbilities concept. The latter will require having a concept FieldsBetweenDiagonal similar to FiledsBetween, however, here the fields shall be on the same diagonal. The latter can be achieved by defining EmptyIntervalOnDiagonal concept analogical to EmptyInterval.

The advantage of such an approach is that the defined units can be reused extensively. For example, once having BishopAbilities and RookAbilities defined we can define QueenAbilities concept just through three simple concepts. We will need to define a QueenAbiities virtual abstract, then define two specifications, where one of them will use FieldsBetween and the other FieldsBetweenDiagonal abstracts.

2.4. Analysis of Improvements

Many optimizations are applied on the Graph of Abstracts to avoid additional nodes creation, which itself brings avoiding additional rules checking, thereby, *speeding up the matching procedure*. Bellow we will report our analysis of naive and optimized models of the Graph of Abstract.

a) *Without optimization:*

The definition requires 4 nucleus nodes, 5 nodes per Field, Figure and each type of Figure, EmptyInterval set abstract requires 6 nodes (so does EmptyIntervalOnDiagonal), FieldsBetween requires 1+2x5=11 nodes (same as FieldsBetweenDiagonal), while each of the first four specifications require 17 nodes and the other four - require 11 nodes each. RookAbilities requires 2 nodes.

In summary, the definition of all figures requires 4+(6+1+1)x5=44 (including the Empty Field), to define FieldsBetween with its specifications 11+4x(17+11)=123 nodes are necessary, and to define RookAbilities - 2 nodes.

b) *With optimization:*

Definitions start with defining 4 nucleus nodes. Here, Field creates only 1 new node and reuses the nucleus ones. Figure creates 3 new nodes (two of the nodes are specializations of FigureType and FigureColor nucleus abstracts). Pawn creates 3 new nodes (on the contrary to other figures, pawn has to specialize both FigureType and Y nucleus attributes),

each other type of Figure (e.g. Rook) creates 2 new nodes - one for the specialized figure type node and the other for the target node. To define EmptyInterval 3 new nodes are created, for definition of FieldsBetween virtual abstract 1 more new node is necessary, and each of the first four specifications require 2 and the other four only 1 new node. RookAbilities creates only 2 nodes.

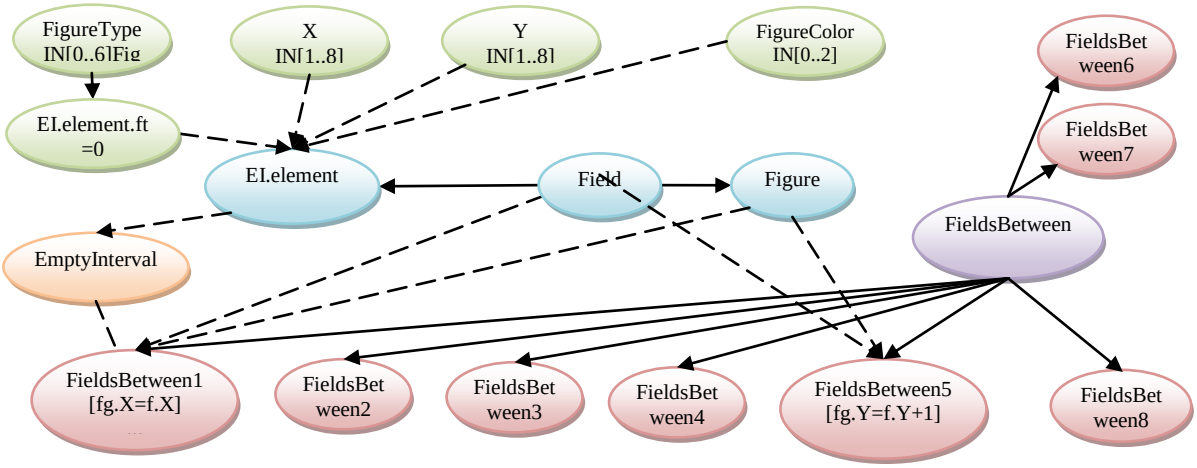

Fig 2. The representation of FieldsBetween and helper concepts in the Graph of Abstract. Some of the nodes and relations are removed from this picture to simplify the view. Solid lines represent *be* and dashed lines *have* connections.

In summary, to define all types of figures only 4+1+3+3+5x2=21 nodes are necessary, while to define FileldsBetween 1+4x(1+2)=13 nodes and to define RookAbilities only 2 nodes are necessary.

What follows is that with optimization we reduce the number of created nodes, therefore the conditions to be checked, for about an order.

3. PRESENTATION OF KNOWLEDGE IN MANAGEMENT

As a management problem we consider the Supply Chain Management (SCM) problem, which is concerned with planning and coordinating the activities of organizations across the supply chain, from raw material procurement to finished goods delivery [12].

3.1 Presentation of TAC SCM Knowledge

Here we will concentrate on the description of the presentation of the relevant knowledge units into the GA.

The main nucleus abstracts for TAC are: Date, Price, Quantity, Speed, ComponentType, ComponentQuality, Brands, Factory capacity, Factory utilization, Bank account, etc. More complex (composite) entities are: Assembled product, Particular supply component, Bank, Factory, Agent, Customer, Supplier, etc. [12]. This type of knowledge is represented in GA exactly like we described for chess.

However, a certain interest represents the specification of Moves and Strategy Plans in TAC SCM. Strategy Plans (SP) for TAC specifies the qualitative changes of some operational parameter(s). In the frames of TAC problem, we can separate two concepts from each other: Move and Strategy Plan (SP). We call as "Move" a statement which contains entities with all their parameters specified with exact values and defined applicable action(s) over the given entities (an equivalence to "move knight" in chess). In opposite, we call as "SP" a statement, composed of one move or set of moves, in which at least one parameter within a single move is not specified with its value. Again, a TAC move describes an exact (quantitative) action over TAC entity (entities), whereas an SP describes qualitative action(s). In GA the formers are defined through Action entities, while the latters are described by *virtual* Actions.

In case of move the Agent directly executes the statement.

In case of SP (with missing one or more parameters' values) the Agent may generate several moves from the same SP by substituting the missing values with ones from their acceptable value range. In order to select the most optimal move from the generated set it applies a game subtree for that appeared intermediate problem with possible usage of the existing knowledge base.

A typical example of TAC move would be the following statement: "Buy N quantity of Queenmax 2GB RAM by D date" [12]. Here we have 3 predefined entities: "Quantity", "Queenmax 2GB RAM" supply component and "Date".

In this example there is an action – "Buy". Here in TAC we should describe action objects with further addition of their implementations into each entity it may be applied to. In the case above the "Buy" action should be described in general and its appropriate implementation should exist for "Queennmax 2GB RAM" entity. That action may also take "Quality" and "Date" entities as mandatory (the first) or optional (the second) parameters. If no "Buy" action is defined for "Queennmax 2GB RAM" entity then that statement loses its meaning and that move or SP will take no effect. "Buy" can, as well, be defined for other type of entities, like "Pintel CPU 3Ghz".

Fig 3 demonstrates the presentation of virtual and non virtual

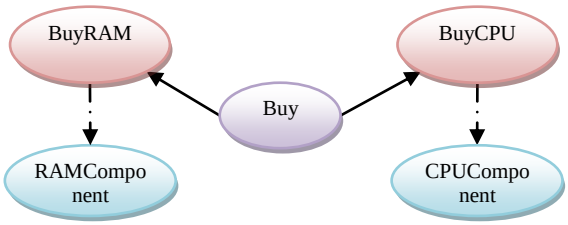

Fig 3. Presentation of Buy virtual action and its specifications in GA. Solid lines represent *be* and dot-dashed lines *do* connections. The specification details of RAMComponent and CPUComponents are not provided to simplify the view.

Actions in the GA. It can be observed that the virtual Actions, like virtual abstracts do not have direct do connections to the other nodes of the GA.

4. PRESENTATION OF INTRUSION PROTECTION KNOWLEDGE

The knowledge for Intrusion Protection problem as defined in [21] is represented in the form of goals and rules. Both of them are easily represented within our model. Particularly, the goal states are defined by the combination of certain important variables: the processor time, buffers size, number

of incoming packages. These basically compose the input nucleus types, and their combinations are defining the critical and normal (or goal) states.

5. EXPERIMENTS WITH MATCHING ALGORITHM

Once we have the GA constructed, we can match situations to the GA and find activated concepts. The matching algorithm is described in details in [3]. Here we consider the activation process and report the results of the following two experiments:

a) Measure the following numbers when keeping the situation unchanged and increasing the number of concepts:
   I. number of checked conditions (NCC)
   II. number of active partial instances (NAPI)
   III. number of active instances (NAI)
b) Measure the number of checked conditions when changing the situation (atomic changes) against the same fixed set of concepts.

As a static situation we will consider the simple situation described in Fig 4 and as concepts we will consider the iterative development of RookAbilities concept, thereby, we will consider 3 separate phases: a) only nucleus abstracts and field/figures, b) adding FieldsBetween with all its specifications, c) adding RookAbilities.

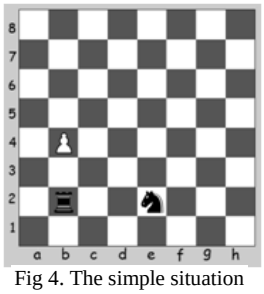

Fig 4. The simple situation to demonstrate the matching procedure.

As we described before, the situations are translated into the lists of instances of nucleus abstracts. Thereby, all fields of the chess board are translated into the groups of nucleus instances which completely define its content. For example, the value of the field containing Pawn in Fig 4 is: {"groupid":34,"instances":[{"type":"X","value":2}, {"type":"Y","value":4},{"type":"FigureType","value":"1"}, {"type":"FigureColor","value":"1"}]}.

Let's consider the matching of the concepts from the first group. The algorithm iterates over the nucleus instances of the situation and fires the instances to the nucleus nodes of corresponding types. In this case it starts from X instance (value is 1) with groupid = 1. It finds X nucleus type node in GA and checks if the value of the instance satisfies the regulations of the node. As the value is 1, it is fired forward. It also creates partial matches for Field, Figure, Rook and other types of Figures. Next comes Y = 8 instance of the same groupid = 1, it is being passed to Y node, which itself adds its instance in the partial matches of Field, Figure, and all types of Figures, except Pawn (Y=8 doesn't satisfy the rule defined in Pawn), with the same groupid. Similarly FigureType of the same groupid fires instance of FigureType nucleus node and registers its instance to partial match Field with the same groupid. Finally, FigureColor instance is being fired, which makes partial match of Field complete, thereby, leads to firing of an instance of Field forward. Similar procedures are applied to the rest of the fields. The summary is given in Table 2.

Table 2. The number of checked conditions, active instances and partial active instances for phase one concepts.

Type	Description	Sum
NCC	64x4 (dispatching)+64 (conditions in Pawn.Y node)+64x(1+7) (conditions for FigureType per each figure - including dummy figure)+64x4x(1+1+7) (conditions to create instances in field/figures).	3136
NAI	64 (Field) + 3 (Figure) + 61 (DummyFigure) + 1 (Pawn) + 1 (Rook) + 1 (Knight) + 64x4 (all nucleus instances)	387
NAPI	61 (Figure) + 3 (DummyFigure) + 63 (Pawn) + 63 (Knight) + 63 (Rook) + 3x64 (the rest of figures)	509

Now, let's consider the processing of the second phase concepts. We shall note, that the matching algorithm works in depth first fashion, thus, once the nodes are activated they are fired further and lead to partial/complete matches of successive nodes.

The activation of FieldsBetween is triggered by the activation of one of its specifications. Let's first discuss the activation of FieldsBetween5... FieldsBetween8 nodes (these are a bit simpler than the first four). As was described before, they do not add any additional nodes but rather are connected to Field and Figure nodes. Thereby, the instances fired from Field or Figure nodes are directly captured by them. Hence, the number of input instances is only 64+3=67. Each of the instances leads to a creation of a partial instance and all possible combinations are considered.

On the other hand, the first four specifications use also an additional concept: EmptyInterval, we shall note that the Set nodes are not activated as free nodes rather they are activated in a backward chaining fashion - once the conditions of the Set's attributes are evaluated. This helps to dramatically decrease the explosive number of instances created by a Set node. The analysis is given in Table 3.

Table 3. The number of checked conditions, active instances and partial active instances for phase two concepts.

Type	Description	Sum
NCC	3x64x8 (all pairs of input instances for each of 8 specifications) + (5+5)x64 (per each satisfying Figure +Field combination being on the same line, we group the satisfying fields in EmptyInteraval concept)	2176
NAI	4 (the neighbors of Rook) +1 + 2(other specification with fields between)	7
NAPI	4+3 (the Rook with Fields where EmptyInterval's conditions are not satisfied)	7

The last analysis of this experiment is to consider the activation of phase three concepts given in Table 1. Here the

Table 1. The number of checked conditions, active instances and partial active instances for phase three concepts.

Type	Description	Sum
NCC	7 (only fired instances of FieldsBetween node are checked)	7
NAI	7	7
NAPI	0	0

Rook abilities work as a filtering node.

What follows from the experiment, is that each successive layer reuses the achievements of the previous ones.

In our next experiment we will consider an atomic change of the initial situation given in Fig 5, and will report the number of condition checks triggered by the change. The aim of the experiment is to show that the subsecutive situations lead to only a few new condition checks.

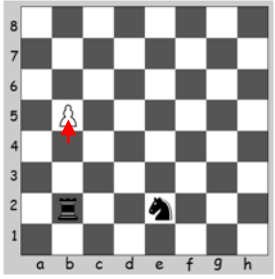

Fig 5. The change in the initial situation.

In the given situation only 4 nucleus instances are changed, they are the FieldColor and FieldType of 34th and 26th groups. All partial and active instances

using these nucleus ones are deactivated. The re-activation analysis is given in Table 4.

Table 4. The total number of checked conditions, active instances and partial active instances triggered by the situation change.

Type	Description	Sum
NCC	4 (nucleus instances) + 4x(1+1+7) (conditions to create instances in field, figures) + 2x64 (for checking one of the modified specifications of FieldsBetween) + 2 (filtering conditions in RookAbilities)	170
NAI	1 (Pawn) + 1(DummyFigure) + 2 (Field) + 2 (FieldsBetween) + 2 (RookAbilities)	8
NAPI	1 (Pawn) + 1 (DummyFigure) + 5 (other figures) + 3 (FieldsBetween)	10

6. CONCLUSION

In the paper we demonstrated the viability of the developed, language syntax based, be-have-do meaning presentation model.

We showed that the model is able to present the knowledge of management, chess and Intrusion Protection SSRGT problems. This allows us to claim that the developed model is able to present the knowledge units of any SSRGT problem.

On the example of the simple Rook Abilities chess concept, our analysis of the optimizations applied in the model showed that the number of created nodes in GA was reduced for about a factor, which saves not only a space but also significantly improves the speed of the matching algorithm.

The analysis of the impact of iterative matching of meanings showed that the activation of each next level consumes significantly less time than the previous one, thus, emphasized the efficiency of the developed model.

Further measurement of the matching algorithm against atomic changes of the situations justified the caching of partial matches. The overall number of checked conditions upon a change, consequently the time spent, was about 30 times less than the number of checked conditions against entire situation.

In addition, for the management problem, represented here by TAC SCM model, the management-specific knowledge units were represented by the described nucleus and composite entities with applied be-have-do model. These knowledge are used during the optimal move search process within a given strategy plan (with, perhaps, their further optimization as the search process goes on), which gradually reduces the calculation volume and used resources.

ACKNOWLEDGEMENT

We would like to express our gratitude to Professor Edward Pogossian for supervising the work.

REFERENCES

[1] Pogossian, E. "Effectiveness Enhancing Knowledge Based Strategies for SSRGT Class of Defense Problems". *NATO ASI 2011 Prediction and Recognition of Piracy Efforts Using Collaborative Human-Centric Information Systems.* p. 16, Salamanca, Spain, 2011.

[2] Pogossian, E. "Combinatorial Game Models For Security Systems". *NATO ARW on "Security and Embedded Systems*, pp. 8-18, Porto Rio, Patras, Greece, 2005.

[3] Khachatryan, K. and Grigoryan, S. "Java Programs for Matching Situations to the Meanings of SSRGT Games". *Proceedings of SEUA Annual conference*, pp. 135-141, Yerevan, 2013.

[4] Pogossian, E. "On Modeling Cognition". *Computer Science and Information Technologies (CSIT11),* pp. 194-198, Yerevan, Sept.26-30, 2011.

[5] Tumer, D., Shah, M.A. and Bitirim, Y. "An Empirical Evaluation on Semantic Search Performance of Keyword-Based and Semantic Search Engines: Google, Yahoo, Msn and Hakia". *The Fourth International Conference on Internet Monitoring and Protection, (ICIMP 2009)*, pp. 51 - 55, Venice, 2009.

[6] Singhal, Amit "Introducing the Knowledge Graph: things, not strings", *http://googleblog.blogspot.co.uk/2012/05/introducing-knowledge-graph-things-not.html*, May, 2012.

[7] Lin, Thomas, et al. "Active objects: actions for entity-centric search". *Proceedings of the 21st international conference on World Wide Web*, pp. 589-598, Lyon, France, 2012.

[8] Uchida, H., Zhu, M., D. Senta, T.G. "Universal Networking Language", UNDL Foundation, International Environment House, 2005/6.

[9] Khachatryan, K. and Grigoryan, S. "Java Programs for Presentation and Acquisition of Meanings in SSRGT Games". *Proceedings of SEUA Annual conference*, pp. 127-135, Yerevan, 2013.

[10]. Louwerse, M., et al. "Cognitively Inspired NLP-Based Knowledge Representations: Further Explorations of Latent Semantic Analysis". *International Journal on Artificial Intelligence Tools*, pp. 1021-1041, 2006.

[11]. Pogossian E., Hambartsumyan M., Harutunyan Y. "A Repository of Units of Chess Vocabulary Ordered by Complexity of their Interpretations". *National Academy of Sciences of Armenia, IPIA*, pp. 1-55, Yerevan, 1974-1980.

[12]. Baghdasaryan, T. "Adapting RGT Solver Interface to Management Strategy Search Problems". *Mathematical Problems of Computer Science*, Vol. 39, pp. 135-145, Yerevan, 2013.

[13]. Pogossian E., Javadyan A., Ivanyan E. "Effective Discovery of Intrusion Protection Strategies". *Workshop on Agents and Data Mining*. pp. 263-274, St. Petersburg, Russia, 2005.