On Systemic Classifications and Machine Learning

Edward Pogossian
Division of Computational and Cognitive networks, IPIA, NAS RA
Yerevan, Armenia
e-mail: epogossi@aua.am

Sedrak Grigoryan
Division of Computational and Cognitive networks, IPIA, NAS RA
Yerevan, Armenia
e-mail: addressforsd@gmail.com

Nairi Hakobyan
Mathematical and programming security of computers, complexes and computer networks , RAU
Yerevan, Armenia
e-mail: hakobyannairi@gmail.com

ABSTRACT
Positioning of a/scmds systems with respect to parametric and neural network models.
A/scmds systems are ontological, systemic, comparable with natural languages in covering mental systems and are constructively modeling mental systems, at least in a fuzzy way.
The questions arise to the abilities of computers in accepting, disposing and properly processing a/smds models.
Some prospective answers to above questions for RGT problems are outlined.

Keywords
Mental system, Systemic, Ontology, Neural network, Combinatory.

1. INTRODUCTION
1.1. We can operate with realities, i.e., everything that causes imprints in us, while, in fact, we operate, at least consciously, with those realities that we can classify.
Utilities are realities that are supporting our goals.
Mental systems represent realities and utilities in particular, but have varying effectiveness with respect to the goals and are processed to support utilization and gaining the benefits from the utilities.
All classified realities, in principle, can model each other. For example, realities themselves provide models of the universe, utilities – the models of realities that support our goals, mental systems are, in particular, the models of utilities.
Classifying mental systems is effective with respect to our goals to the extent to which they provide appropriate utilities regularly, and are modeling the utilities constructively and adequately.
1.2. Natural languages, parametric, statistical and other types of algorithms, programming languages, logical calculus, are systemically modeling mental systems in a wide range of constructiveness and adequacy [32].
Natural languages (NL) of C are systemic and comprehensive by their coverage of mental systems of C but classifiers NLCl of NL are not constructive and are modeling MsCL only fuzzy because determine not the positives of +MsCl but only IDs of positives and IDs of relations between them.
There are several other classifiers providing constructive, adequate, or both, models of subclasses of MsCl. Particularly, classifiers in sciences, algorithms in a variety of equal modes, say modes of programming languages (PL), parametric, statistic and combinatorial algorithms, methods and deterministic methods, logical calculus's, others.
Algorithms, for example, are constructive and systemic since comprise certain systems of regs/rules but represent only a part of mental systems, namely, mdoers.
OOP PL are covering mdoers as well but are more systemic with respect to algorithms since involve attributes/parents/doing, or Have/Be/Do in [13, 32], rels.
Then, predicate calculus is constructive but systemic only partly with respect to MsCl since include only rels of logical connectivity, commonality and existence.
1.3. In [32] constructive models of mental doers and systems of those doers, cmds and a/scmds, are specified.
A/scmds systems are ontological, systemic, and comparable with natural languages in covering mental systems and are constructively modeling mental systems, at least in a fuzzy way.
An evidence to support the adequacy of the a/scmds modeling of Explaining, Understanding and Human-Computer Communications were provided in [32].
Cmds by construction are consistent with algorithms by Markov and basic units of OOP, thus, following Church cmds are adequately modeling, at least, Computable Functions (CF).
1.4. Mental systems specifying other mental systems, in turn, can be questioned to be specified and modeled, assuming certain classifying them mental systems were already acquired from thesauruses of scientific communities.
Apparently, this chain can be continued.
Some modes of the above questions are studied in the branch of theory of algorithms, synthesis of algorithms, where assuming a priory certain classifiers of mdoers are already given algorithms of synthesis of equal constructive versions of those mdoers are developed.
Deductive modes of synthesis those mdoers can include certain axioms and logical statements or can be determined recursively [26].
In the inductive modes, including machine learning, those mdoers can be represented by samples of their domains or their representations, performances of mdoers, others [39].
1.5. Certain mental systems provide methods of transmission, teaching of mental systems inside of communities C as well as methods of acquisition of those mental systems.
While commonality of thesauruses of members of C let them avoid specification of those methods it becomes unenviable in transmitting and acquiring human mental systems by computers.
In contrast with machine learning where teachers are forced to provide to computers the representations of mental systems step by step, by portions, teachers can do that holistically and completely when they teach them. The questions arise to the abilities of computers in accepting, disposing and properly processing those mental systems.
Prospective answers to above questions for RGT problems are presented in [13, 15-17, 27, 28, 29, 30].
1.6. In IIAP since 1957 chess and combinatorial problems have been studied. A class of combinatorial problems where space of solutions are reproducible game trees (RGT) are considered.
The RGT is a spacious class of problems with only a few following requirements to belong to- 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

- the situations the actors act in and transformed after the actions can be specified by certain rules, regularities.

Many security and competition problems belong to RGT class. Specifically, these are network Intrusion Protection (IP), Management in oligopoly competitions and Chess-like combinatorial problems.

Unified RGT specification of problems makes it possible to design a unified Solver for the problems of the class.

Solver of the RGT problems is a package aimed to acquire strategic expert knowledge to become comparable with a human in solving hard combinatorial competing and combating problems.

We formulate the limitations in designing effective package as follows:

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

Overall RGT Solver has several main components. It consists of graphical user interface, which provides tools and structures for doers' insertion, controller, which handles mdoers, creates nodes from them and stores in a network, called Network of Abstracts, does situation processing with matching algorithms. The last module is PPIT (Personalized planning and integrated testing) module which stores plans and goals and searches for strategies based on plans

1.7. In this paper we are positioning a/scmds models with respect to known parametric, statistic, neuron nets (NN) models preliminarily outlining those models.

Then address to computer realization and experiments with systemic classification, acquisition of and matching to mental systems in frame of RGT Solver to conclude with outlining our basic statements.

2. PARAMETRIC, NON-PARAMETRIC, NEURAL NETWORKS CLASSIFICATIONS AND MACHINE LEARNING

2.1. Machine learning algorithms can be classified by different modes – supervised/unsupervised; parametrical (statistical)/non-parametrical, etc.

For positioning let's outline some of these methods, describe some advantages/disadvantaged and possible use of machine learning algorithms by parametrical/non-parametrical classification.

2.2. Parametrical methods.

Description, advantages: the main algorithm of Parametrical Machine Learning includes 2 steps – selecting a form for the function and learning the parameters/coefficients for the selected function. Some examples of PML algorithms are regression (linear, polynomial, ridge, logistic, etc.), LDA, etc.

PML has some advantages in comparison with NPML (also some disadvantages which would be described in the next subchapter), let us mention some of them, these methods:

1. are easy to understand
2. are learning from data very fast
3. need less training data.

Problems: wrong choice can impact on problems like overfitting/underfitting, but there are many popular methods to avoid it. [19]

"All models are wrong, but some are useful" © [20]

This quotation by George Box in 1976 has some similarity to a popular theorem called No Free Lunch. The difference is – Box means that NO model can solve (predict) by 100 % accuracy, and even if it reaches 99 % it cannot be considered as right. Anyway, Box also mentioned that some models are useful, meaning that it's useful to solve some problems by some (not 100) percent of accuracy and a little (maybe) error.

Nowadays, ML algorithms are a big part of business, so reaching 80 % of needed accuracy means 80 % of profit, in most cases.

No Free Lunch theorem consists that no model works best for every problem and it's intuitively clear.

Let's mention also some limitations that may appear while using PML algorithms:

It was mentioned in the list of advantages that PML algorithms are simple to understand. One of the points of it is that they are better suited to simple problems; consequently they have limitations on problem's complexity;

The solution/prediction highly depends on the chosen functional form, say, choosing a polynomial regression of degree 2 can give a much better result than the same regression of degree 3 and much worse than degree 4. High fluctuations create problems.

Statistical methods

Statistical Machine Learning (or parametric statistics) is a part of PML. Parametric statistics is a branch of statistics which assumes that sample data come from a population that follows a probability distribution based on a fixed set of parameters.

The difference is that parametric methods put general issues, while statistical methods are used when we need to retrieve a distribution and find parameters.

2.3. Non-Parametrical methods

Description, advantages: talking formally, algorithms that do not make strong assumptions about the form of the mapping function are called nonparametric machine learning algorithms.

There are many popular NPML methods like Decision Trees, k-Nearest Neighbors (taking only k closest training data examples), and Bayesian Image Analysis, etc.

This type of algorithms does not insist using no parameters, but that number is flexible and usually increases while learning from data. This can be considered as one of the advantages of NPML in comparison with PML.

The other advantage, as already mentioned above, is that NPML does not have assumptions about the mapping function.

Problems: let's now list NPML disadvantages:

a) slow working – it's intuitively clear – if algorithm allows changing of mapping function while reading more and more data it would work slower

b) more risk to have overfitting

c) requires much data –> more data –> more changes in mapping function -> better results

2.4. Neural Networks

Description, advantages: NN has very different types and usages [21] [22] [23], anyway, we cannot describe them all, so we will discuss them on the whole.

NN has a lot of advantages but also some big disadvantages. One of the biggest advantages is organic learning. This means that NN outputs aren't limited by inputs. NN has the ability to generalize its inputs.

In real life, knowledge can be unitary or composed into a system of units, so when presenting knowledge we divide it into components and describe them separately (each of the components may be also divided, etc.) and define relations between them. Say, if we want to describe the room's condition, we should do it in the mentioned way.

In this view, NN is closer to human perception – first layers can be for understanding the general shapes, and the last ones for exact shapes needed for the given task, say, in chess, first layers can be used to detect nuclear variables (for

example), somewhere in middle it could be mate classifiers (if possible to find with ML algorithms), etc.
The other advantage is the self-repairing system. If NN is asked to find out specific data which is no longer in communication, it can regenerate large amounts of data and help in determining the node that is not working.
Problems: NN is surely a very powerful instrument, but many companies avoid using it – why? NN usually needs a long time for working, therefore it is not suitable for real time data. For instance, anomaly detection problem (in most cases) is being solved with simpler algorithms – Box-Cox transformations, simple regressions and more but not NN. The problem is that anomaly is needed to be detected fastly while NN is important for slower work.
Some other NN issues are: the selection of the network model, the pre-processing of the information for training data formation, etc.

2.4.1. Deep Learning

Deep learning is a part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms.
Deep learning architectures such as deep neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, where they produced results comparable to human experts [24][31].
Deep Learning is a powerful set of techniques for learning in neural networks. It has huge advantages:

1. Deep Learning can account for different types of ambient conditions, like product reflection or lens distortion, and learn interesting features to make an inspection robust, so it identifies defects that are difficult to detect by other methods.
2. ML has a problem in finding irregular shapes or patterns that do not have any symmetry so product variation can render the traditional ML impractical.
3. Deep Learning can detect more subjective defects that are difficult to train such as minor product labeling errors like incorrect fluid ounces or region that would relate to a significant recall.
4. It is an architecture that can be adapted to new problems relatively easily (e.g., Vision, time series, language, etc.), using techniques like convolutional neural networks, recurrent neural networks, long short-term memory, etc.

Let's also describe some disadvantages:

1. Deep learning makes better performance if the amount of data is huge. If data is little then older learning algorithms will outperform deep learning.
2. Cost problem – the most complex models may take days or weeks to train using big amount of machines.
3. What is learned is not easy to comprehend. Other classifiers (e.g., decision trees, logistic regression, etc.), make it much easier to understand what's going on.

To summarize, some different methods of ML were described with both advantages and disadvantages listed. Anyway, it should be understood that natural language is the most extensive coverage as a model for all beings, but the language has shortcomings, and the systemic classifications try to eliminate defects and at the same time present the whole coverage as maximum possible.

3. POSITIONING CONSTRUCTIVE SYSTEMIC MODELS

3.1. Systemic Models

Members of communities C understand explanations of mental systems m of members of C if, ideally, activate their own equal to m mental systems m'.
Combinatorial problems, particularly chess, are being solved with mapping the essence of the system into parameters with parametric and machine learning approaches.
In [3], as well as in [4, 5] it is stated that the combinatorial nature of those problems and their situations (particularly chess), cannot be adequately averaged into parameters in principle. The line of those researches looks for systemic approaches for solving those problems.
Using mental systems in systemic approaches as a base is successfully applied to various problems, including combinatorial games, planning in battlefields [6], marketing problems, etc., projects, such as SOAR [7] are being developed for general planning and strategy searching problems within this line.
There are various ways of presentation of mental systems, rule-based [8] approach describes mental doers with rules representing "if <A > then <A>" schemes, the advantage of such approach is that it is quite easy to define this kind of mental doers and it already represents a strategy, however, it has disadvantages such as difficulty of induced decisions, dynamic updating and reuse of rules.
Ontologies represent high level properties, relations in mental systems, however, ontologies are not flexible enough to represent all the space of mental doers in the system (such as plans), and cannot be matched.
UNL (universal networking language) [9, 10] is a declarative formal language specifically designed to represent semantic data extracted from natural language texts. In the UNL framework, the information conveyed by natural language documents is represented by a semantic network, i.e., a network which represents semantic relations between mental doers. This semantic network, or UNL graph, is made of three different types of discrete semantic entities: Universal Words, Universal Relations and Universal Attributes. Universal Words, or simply UW's, are the nodes in the semantic network; Universal Relations are arcs linking UW's; and Universal Attributes are used to instantiate UW's. UNL tries to encode the meanings of natural language sentences with a detailed representation of the connections between words it lacks the possibility to automatically associate the words to the realities. In other words, the mechanism of the situation recognition is not developed there and nodes represent only relations, not regularities inside the objects and its attributes.
Other systemic approaches include combined approaches.
Google constructed Knowledge Graph [11] to enhance its search engines with systemic solutions, where it uses knowledge databases.
Widely used OOP languages, UML are also systemic representations of problems [12].

3.2. a/scmds models

3.2.1. An essential requirement to the a/scmds systemic models in [30] is their consistency with the basic acknowledged classifiers of mental behavior, doings as follows.
3.2.2. Humans, at least, since the birth do to benefit from utilities, i.e., from realities directly or not favoring to their roots, as well as to avoid from damagers, to utilize realities either already classified as reducible to utilities or to classify uncertain, yet, ones [30].
The essence of gaining memberships in communities C is in acquisition of accumulated by C and commonal in C doers, certain meta doers of controlling and developing them as well as communicatives of those doers to communicate about realities aimed to coordinate the efforts of members of C in solving common in C problems.
3.2.3. For humans those doers are mainly mental ones while communicatives of mental doers can be their IDs or some

representations of their nature, say samples they classify, or models of those samples.

In [30] it was assumed that mental doers comprise mental systems having unique IDs and united in nets with nodes and relationships between them assigned by those IDs.

3.2.4. Mental systems in a variety of scopes are intrinsic for humans and in a variety of life time periods are formed genomic or cognitively.

They are backing doings for roots, say consuming energy and matter of others, or goals induced by roots and varying in effectiveness with respect to them.

Communicatives of C of certain types represent mental systems and can be organized into languages L, like English is organized from IDs of certain mental systems that have thesauruses comprising mental systems of C, corpuses of the totalities of IDs, syntacies and semantics of L [30].

Mental systems in the nets of mental doers correspond to connectivity subnets rooted in the nodes of subsystems of connectivity subnets.

Mental systems determine classes of equality of realities, their nodes can be processed for several goals, can be decomposed or abstracted.

For example, mental systems Factories include nodes of their stuff, workshops, buildings, etc., and a variety of mental systems and realities can be equal or match them, be decomposed or abstracted.

Other aspects in classifying systems include, particularly, synchronous availability out of their origin in time , say in past or in present, being causes or their effects,

and equidistant accessibility like the nodes in "star" types networks.

3.2.5. Effectiveness of mental systems rises by, particularly, grounding msystems to commonly acknowledged ones like axioms or rules of logical inferences, eliminating malicious circles in their representations , and more, being consistent with , say, cause –effect or originations in time chains of realities the mental systems represent.

The most powerful mental systems, seemingly, we gain when provide their constructive models and make them adequate [30].

4. CONSTRUCTIVE SYSTEMIC MODELS FOR RGT SOLUTIONS

4.1. Network of abstracts

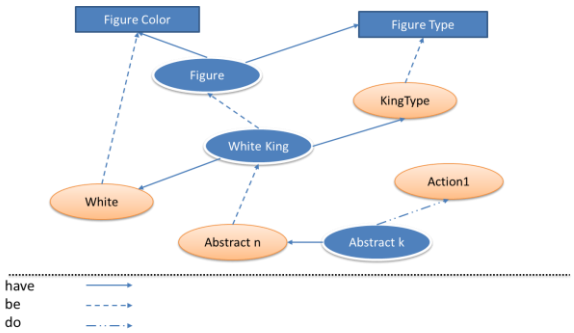

Fig. 1 Network of Abstracts and HBD relations

We construct Network of Abstracts (NA) based on mdoers defined via interface. Their Structure is based on English language grammar, particularly Have, Be and Do (HBD) main dimensions of its verbs [13, 14]. Similar to classes in OOP, say in Java, HBD model of the mental system presentation enables abstracts in NA related with Be (inheritance) and Have (attributes) relations, as well as Do relation (similar functions and methods of OOP) are achieved by actions. There are 4 types of abstracts identified: 1) Nuclear/primitive abstacts which serve as ground for construction of NA, particularly for chess they can be Figure Color (black, white), Figure Type (king, queen, rook, etc.), X and Y cords (to define the chess situation on the board) 2) Composite abstracts which can be composed of other types of abstracts, can be derived from other composites and can be virtual (e.g., check concept of chess is virtual and its specifications are check by knight, check by pawn, etc.), 3) Sets, which are similar to arrays in OOP, enable definition of bundles of abstracts of the same type 4) Actions which consist of composite precondtions and regularities defining situation updates [14].

Fig. 1 shows the Network of Abstracts, where all 3 types of HBD relations are presented.

The matching of situations to abstracts in NA is performed as described in [15]. In RGT Solvers abstracts and situations are stored in store of abstracts (Fig. 2).

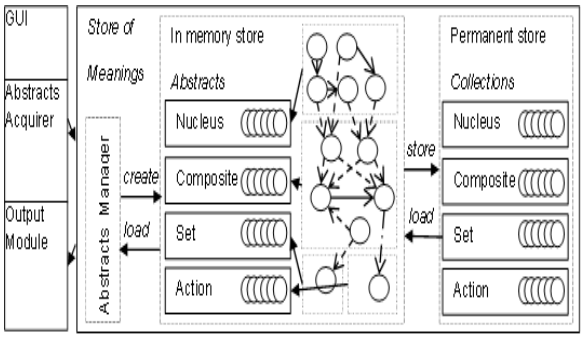

Fig. 2 Store of abstracts

Solver's processing of situations iteratively triggers the matching or ablation of the instances of abstracts lying in the Network of Abstracts, consequently, building the matched set of abstracts in the Solver. The procedure of composing abstracts from sub-abstracts represents a classical constraint satisfaction problem. NA nodes are divided into two functional categories: filtering and conjunction nodes. The first ones decide whether to discard or propagate further matched instances by applying conditions. While, the conjunction nodes serve as assemblers trying to match different sets of matched attributes' instances for generating a new instance corresponding to the node's abstract.

Matching algorithm description

Nucleus nodes serve only as filters, Composite nodes are combining the characteristics of both node types uniting the instances of attributes and filtering out the ones which violate the rules. It is necessary to check all possible combinations of the attributes which are achieved by keeping the list of partial instances which represent all allowed (by rules of the abstract) combinations of already arrived sub-instances. Set nodes are just collecting the same type elements satisfying the rules described in [16]. Action nodes are considered matched when their precondition is activated, which is represented as a composite node.

The matching to virtual abstracts is done as described: they are being matched by their children (called specifications of virtual abstracts) when the latter ones get matched, when they are used as attributes in other abstracts (those are called usages) they are activated by their parent virtuals.

The presentation and matching also support negation of abstracts [16].

4.2. Structure of Plans and Goals

Strategies in RGT Solvers are constructed based on goals and plans [17, 18].

In Solver goals consist of precondition, postcondition which are composite abstracts depth of goal search tree. Preconditions are situations for which this goal is applicable. This basically defines the pattern of situations where goal is meaningful. For some goals the precondition can be any situation. Postconditions are situations which appear when the goal is achieved, e.g., if the goal is "make check with the queen", after it is achieved the opponent king is under check

of queen in the given situation, this can be any situation as well.

Plans are list of goals sorted with their priorities.

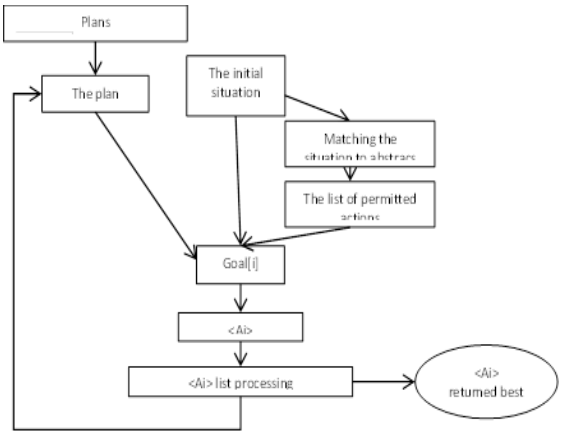

Fig. 3 Block scheme of optimal move searching algorithm

4.2.1. Searching for Strategies based on Plans

Processing of plans for searching strategies on the given situation is done by the goals defined in the plan. Goals are processed and the goal with the highest priority which matches is selected and the optimal moves for it are suggested (Fig 3).

4.3. Let's discuss the situation on the chess board in Fig 4, in Solver it is defined as in Fig 5.

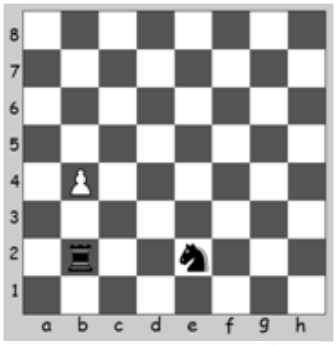

Fig. 4 Chess board example for matching demonstration

4.3.1. First level of matching

Matching starts 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 the corresponding types. In this case it starts from X instance (value is 1) with groupid = 1. It finds X nucleus type node in NA 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

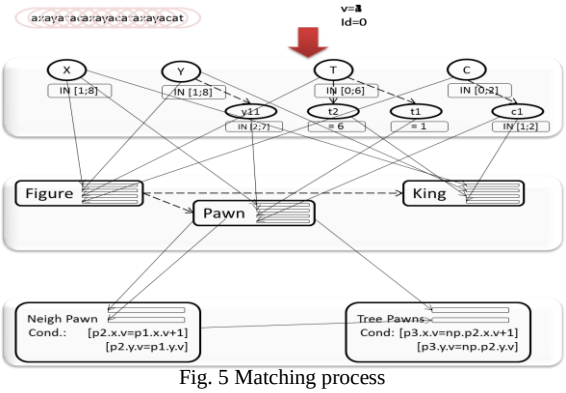

Fig. 5 Matching process

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.

4.3.2. Second level of matching

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 FieldUnderCheck is triggered by the activation of one of its specifications. It has a specification FieldUnderCheckOfRook which is also virtual and is specified as a composite abstract that contains a Set abstract EmptyFields in it which defined the line between two figures, in this case fields between rook and the pawn and fields between the rook and the knight. Let's first discuss the activation of. Sitaution processing over already activated figures and fields triggers activation of 2 EmptyFields with between rook and the pawn and the rook and the knight. Rest fields are not added into those EmptyField type abstracts, because of rules defined in them [16]. On the same level moves by figures are activated as their preconditions are activated, more abstracts activate on the next level of those abstracts, such as FieldUnderCheckOfRook, FieldUnderCheckOfPawn, and other abstracts. Matching to next levels is happening similar to this, when a new node is activated (as said above here abstracts compose of filtering and conjunction functions).


```
{"name":"sit_1","elements":[
  {"groupid":1,"instances":[
    {"type":"X","value":1},
    {"type":"Y","value":8},
    {"type":"FigureType","value":"0"},
    {"type":"FigureColor","value":"0"}
  ]}
  …
  {"groupid":34,"instances":[
    {"type":"X","value":2},
    {"type":"Y","value":4},
    {"type":"FigureType","value":"1"},
    {"type":"FigureColor","value":"1"}
  ]},
  ...
]}
```


Fig. 6 Situation presentation in Solver

What follows from the experiment, is that each successive layer reuses the achievements of the previous ones. As shown in the image the first level of matching is nucleus abstracts, such as coords, figure type and color in the case of chess. Next level is appearing simple knowledge pieces, such as figure and certain figure types, as well as fields in chess. Next it comes the composite level where nodes can have both filtering and conjunction. In the example FieldUnderCheckOfPawn is an abstract matched on that level.

The essence of HBD model is to build a shell which would represent the knowledge using linguistic relations while the same time will enable automatic matching of the situation to the existing knowledge. Thereby, the advances in the UNL are opening perspectives to integrate and enrich the linguistic relations used in the HBD model and implement more advanced matching algorithms.

5. CONCLUSION

Systemic approach to problem solving is discussed. Overview for various mental system models is given, including parametric, statistic methods, as well as neural networks are discussed with their advantages and disadvantages.

Expectations from mental systems are defined and their constructive models are discussed and some of their shortcomings are mentioned.

Our approach for modeling constructive mental systems for RGT class of problems is given with overall abilities overview. RGT Solvers are able to acquire RGT problems and provide systemic solutions to them. Mental doers in RGT Solvers are presented in the Network of Abstracts which is also able to be matched to situations.

We plan to prove the adequacy of our a/scmds models for mental behaviors classified by psychologists and psychiatrists as keys to identifying the wellbeing of humans.

Future development of RGT Solvers is expected to be developed in the following directions: a) Enhance matching algorithms with machine learning solutions, b) enhance mental doers presentation and acquisition with natural language and UNL bases.

REFERENCES

[1] A. Tikhonov, V. Arsenin, "Methods for solving some problems", *Science*, Moscow, 1974.

[2] E. Pogossian, "Adaptation of combinatorial algorithms" (in Russian), p. 293, Yerevan, Armenia, 1983.

[3] M. Botvinnik, "About Solving Approximate Problems", Moscow: Sov. Radio, 1979.

[4] J. Pitrat, "Consciousness and Conscience, in Artificial Beings: The Conscience of a Conscious Machine", *ISTE*, London, UK, 2009.

[5] D. Wilkins, "Practical Planning: Extending the Classical AI Planning Paradigm", *vol. 205, Morgan Kaufmann Publishers Inc.*, p. 205. San Francisco, CA, USA, 1988.

[6] B. Stilman, M. Aldossary, "Revisiting Major Discoveries in Linguistic Geometry with Mosaic Reasoning" *Procedia Computer Science Volume 46*, p. 784, 2015.

[7] J. Laird, "The Soar Cognitive Architecture", *MIT Press*, England, 2012.

[8] B. Buchanan, M. Shotcliffe, "Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project" *Addison-Wesley, Reading*, Massachusetts, 1984.

[9] UNDL Foundation, www.unlweb.net, 2013.

[10] "UNL Specifications", UNL Center of UNDL Foundation, 2005.

[11] A. Singhal, "Introducing the Knowledge Graph: Things, Not Strings", *Official Blog of Google*, 2012.

[12] D. Poo, D. Kiong, S. Ashok, "Object-Oriented Programming and Java", *Springer Science+Business Media*, London, 2008.

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

[14] K. Khachatryan, S. Grigoryan, "Java programs for presentation and acquisition of meanings in SSRGT games", *Proceedings of SEUA Annual conference*, pp. 127-135, Yerevan, Armenia, 2013.

[15] K. Khachatryan and S. Grigoryan, "Java programs for matching situations to the meanings of SSRGT games", *Proceedings of SEUA Annual conference*, pp. 135-141 Yerevan, Armenia, 2013.

[16] S. Grigoryan "Research and Development of Algorithms and Programs of Knowledge Acquisition and Their Effective Application to Resistance Problems", *PhD*, p 111, Yerevan, Armenia, 2016.

[17] E. Pogossian, V. Vahradyan, 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*, pp. 229-241, St. Petersburg, Russia, June 6-7, 2007.

[18] S. Grigoryan, "Structuring of Goals and Plans for Personalized Planning and Integrated Testing of Plans", *Mathematical Problems of Computer Science, vol. 43*, pp. 62-75, 2015.

[19] H. K. Jabbar, R.Z. Khan, "Methods to avoid over-fitting and under-fitting in supervised machine learning", *University of Misan*, Misan, Iraq.

[20] G. E. P. Box, "Science and Statistics", *Journal of the American Statistical Association, Vol. 71, No. 356.*, pp. 791-799, 1976.

[21] O. Matan, R. Kiang, C. Stenard, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, L. Jackel, Y. Le Cun, "Handwritten Character Recognition Using Neural Network Architectures", *AT&T Bell Laboratories*, Holmdel, 1990

[22] M. Auli, M. Galley, C. Quirk, G. Zweig, "Joint language and translation modeling with reccurent neural network", *EMNLP*, 2013.

[23] K. Luk, J. Ball, A. Sharma, "An Application of Artifical Neural Networks for Rainfall Forecasting", *Mathematical and Computer Modelling, Vol. 33, Issues 6-7*, pp. 683-693, 2001.

[24] LeCun, Yann, Y. Bengio, G. Hinton. "Deep learning." Nature 521.7553, pp. 436-444, 2015

[25] K Fu, "Syntactic Methods In Pattern Recognition", p. 322, London, 1974.

[26] H. Marandjian, "A Method of Synthesis of Programs of Numeric Functions", *Mathematical Problems of Cybernetics and Computers vol. 26*, pp. 5-13, Yerevan, 1986.

[27] E. Pogossian, A. Martirosian, "Learning", *Reference book on Intellectual Systems. Radio i Svjas Publishing Company*, Moscow, v.2, pp. 206-231, 1990 (in Russian).

[28] E. Pogossian, "Specifying personalized expertise. International Association for Development of the Information Society (IADIS)", *International Conference Cognition and Exploratory Learning in Digital Age* (CELDA 2006), Barcelona, Spain, pp. 151-159, 2006.

[29] E. Pogossian, "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*, pps. 16, Salamanca, Spain, 2011.

[30] E. Pogossian, "Towards Adequate Constructive Models of Mental Systems", *Computer Science and Information Technologies (CSIT17)*, Yerevan, Armenia, 2017.

[31] Hinton, Geoffrey, et al. "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups" *IEEE Signal Processing Magazine 29.6*, pp. 82-97, 2012.

[32] E. Pogossian, "On a Transparent Presentation of Written English Syntax", *5th Intern. Cognitive Linguistics Conference, Vrije Universiteit*, pp. 209-214, Amsterdam, July 1996.