Mathematical Problems of Computer Science 50, 96--103, 2018.

Experimenting with Acquisition of and Matching to Systemic Classifiers

Sedrak V. Grigoryan and Nairi P. Hakobyan

Institute for Informatics and Automation Problems of NAS RA
e-mail: addressforsd@gmail.com, hakobyannairi@gmail.com

Abstract

We are modeling acquisition and classification abilities for the machine. The research line we follow, is based on the ideas of inventors of algorithms letting constructively model human computations and on some extension of those ideas aimed to model constructively other mental doings [1, 2]. We question the issues of acquisition of and matching to systemic classifiers and experimenting to prove the adequacy of our models. We experiment in the frame of RGT class of combinatorial problems for a RGT kernel problem, chess.

Keywords: Systemic Classifiers, Matching, Chess, RGT, Cognitive Systems, Acquisition.

1. Introduction

We study the nature and the functions of cognition. There are different lines of researches in this area, e.g., machine learning solutions, such as neural networks concentrate on modeling of biological nature of human brain.

We concentrate on studying of the nature of cognitive functions and the ways they are being processed.

Suggested in [1, 2] Theory of Mental Doings provides ways for constructive and adequate models of various cognitive functions, e.g., classification, explanation, etc.

In this work we conduct experiments on developed models of systemic classification theory addressing to described in [3] acquisition of systemic classifiers and matching to them.

1.1 Human in Universe Problems

Human deals with realities, some of which are not classified, while he/she mainly deals with classified realities.

Classified realities can be divided into regularized and not regularized.

A mighty way of enhancement of effectiveness of mental systems, and thus, cognizers, is the regularization of classifiers induced by mdoers and mental systems.

Namely, classifiers Cl of members x of communities C are regularized in C if accompanied by ontological in C methods, instructions allowing x regularly provide positive samples of inputs of Cl as well as let the members of C do the same by communicating with x. [1].

1.2 Constructively Regularized Classifiers

Regularized classifiers can be considered in two spaces: constructively regularized and non-constructively regularized.

In constructive regularization those samples can be provided deterministically and without any involvement of cellular, while, otherwise, can be grown up from a priory given prototypes like cells or crystals, be the products of services to humans or machines.

Regularized classifiers induced can be equally represented, we assume, by (fuzzy) methods, and thus, following Church thesis by (fuzzy) algorithms.

Regularly provided positives r of classifiers Cl and Cl themselves are interpreted as models of classifiers Cl' if r are classified as positives of Cl` and Cl are interpreted as adequate models of Cl' if positives r meet certain additional requirements focused for positives of Cl.

For example, algorithms are adequate models of deterministic methods if, following Church, to any method by certain instructions equal algorithms can be corresponded. [1]

1.3 The Line of Our Research

Interpreting the aims of algorithms to enhance the effectiveness of classifiers of deterministic methods other classifiers focusing the mental ones that extend algorithms are suggested and the following statements were provided [2]:

i. Algorithms are modeling and constructively regularize deterministic methods.
ii. OO Languages are constructively regularized and strongly expand algorithms.
iii. Mentals are constructively regularized and strongly expand OOL.
iv. Mentals can consist of functional and connectivity mental models.
v. For languages L of communities C allowing the members x of C to communicate, i.e., to explain and understand mental systems of each other expressed in L, it can be constructed communication algorithms LC letting computers communicate mental models Mns of mss Ms of C equally with respect to the members of C if Mns and Ms are equal to each other.

Natural languages contain a large number of constructively classified mentals, e.g., English has about 300 000 classifiers. [2]

1.4. Current Work

Following [1,2], algorithms are type of systems constructively modeling computational mental doings over numeric input IDs of realities and OO languages expand them by adding *attributing/have, parenting/be* and *do* relations.

Mentals aim to expand OO languages to systems, exempted from the requirement of only numeric inputs, i.e., allowing with numeric input IDs provided by experts the ones of any given sensors.

While we need to provide ways to construct relationships we identify in natural languages, for now we concentrate on providing main *have/be/do* relations.

We are going to experiment with mentals representing mental systems of RGT problems.

1.5. RGT Class of Problems

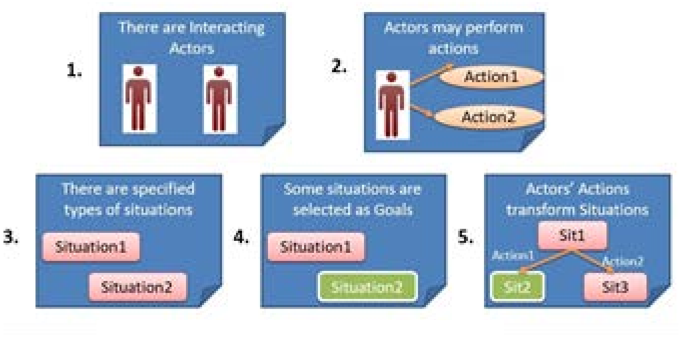

1.
There are Interacting Actors
2.
Actors may perform actions
Action1
Action2
3.
There are specified types of situations
Situation1
Situation2
4.
Some situations are selected as Goals
Situation1
Situation2
5.
Actors' Actions transform Situations
Sit1
Action1
Action2
Sit2
Sit3

Fig. 1. Peculiarities of RGT Class.

To provide a certain assessment, progress and statements we concentrate on a class of combinatorial problems, which is regularized and where space of solutions are reproducible game trees (RGT)[4,5].

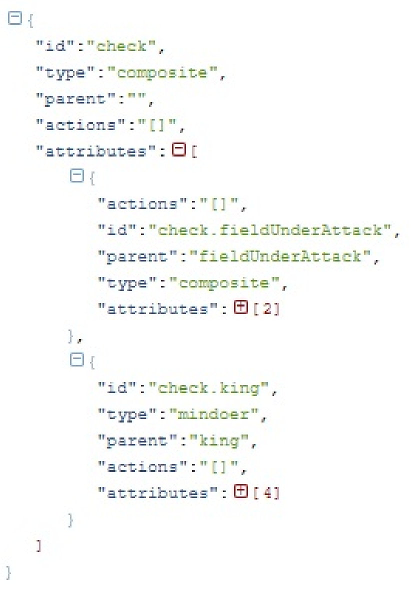
RGT is a class of problems that satisfies the following conditions:

1. 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.
2. There are identified benefits for each of the actors.
3. There are descriptions of the situations the actors act in and transformed after actions.

Chess and chess-like games, network intrusion protection, management in oligopoly competition are considered as RGT class problems. ([4-6]). It has been shown [4] that the kernel of these problems is unique, which lets having unified framework for achievements and experiment solutions and achievements for a certain kernel problem (say chess) and then spread the solution to the whole class.

Thus, in the following work we conduct an experiment developed by RGT Solver, which implements mentals for RGT problems and relevant knowledge. In the previous works models and algorithms for matching RGTs situations were introduced [3], but their adequacy was not demonstrated, hence, in the following work we aim to provide experiments for RGT Solver acquisition and matching functionalities, particularly we are going to experiment 'mate' chess concept acquisition and matching to situations.

2. Acquisition of Systemic Classifiers



```
⊟{
   "id":"check",
   "type":"composite",
   "parent":"",
   "actions":"[]",
   "attributes":⊟[
      ⊟{
         "actions":"[]",
         "id":"check.fieldUnderAttack",
         "parent":"fieldUnderAttack",
         "type":"composite",
         "attributes":⊞[2]
      },
      ⊟{
         "id":"check.king",
         "type":"mindoer",
         "parent":"king",
         "actions":"[]",
         "attributes":⊞[4]
      }
   ]
}
```

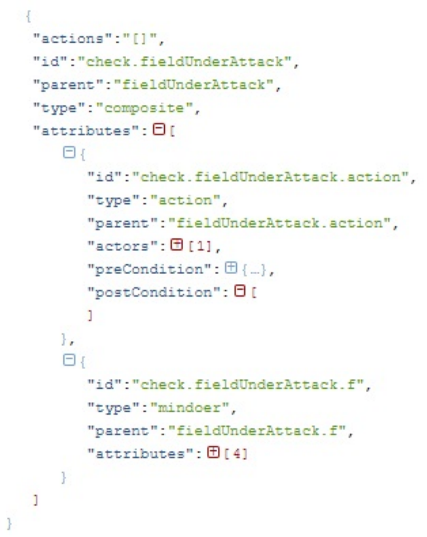
Fig. 2. Classifier 'check' by JSON.

OOP is the most widely spread programming language now. It deals with objects and relations between them, as well as classes of objects. It is close to natural understanding of realities.

We focus on the mental abilities that algorithms or OO languages model poorly or partially. We have a natural language, its description, we want to make the characteristics of a natural language that will be adequate and constructive to language classifiers.

We must test whether we can do the same thing by mentals. We classify our mentals and mental systems in human-space interaction and try to reflect the idea of a human being there. To narrow it down, we took RGT class, where the problem can be considered as a universe, and the solution can be interpreted as a human solution.

For example, the described problem will sound for chess approximately as follows: can we do everything through the RGT class in Solver that we can do through a Natural



```
{
 "actions":"[]",
 "id":"check.fieldUnderAttack",
 "parent":"fieldUnderAttack",
 "type":"composite",
 "attributes":⊟[
    ⊟{
       "id":"check.fieldUnderAttack.action",
       "type":"action",
       "parent":"fieldUnderAttack.action",
       "actors":⊞[1],
       "preCondition":⊞{…},
       "postCondition":⊟[
       ]
    },
    ⊟{
       "id":"check.fieldUnderAttack.f",
       "type":"mindoer",
       "parent":"fieldUnderAttack.f",
       "attributes":⊞[4]
    }
 ]
}
```

Fig. 3. Classifier ‘Field under Attack’ by JSON.

Language? Particularly, can we explain some idea to the Solver, and then expect it to explain it back to us?

Here comes the acquisition problem. The knowledge taught by the teacher has been mastered through experiments. The expert is able to transfer his classifiers regularly to the Solver.

The knowledge is transmitted to Solver by JSON format, which includes classifier ID, type, possible parent ID, its attributes, actions (if any) etc.

Here is an example of input for simple chess classifier ‘check’.

Classifier ‘check’ consists of two attributes/classifiers ‘field under attack’ and ‘king’. Attributes for them are hidden in the picture for better understanding.

Here are inputs for both of them: ‘Field Under Attack’ consists of an action attribute, i.e., ’move figure’ and ’field’ attribute (i.e., where that figure can be moved), and if the color of a figure in ’move figure’ and ’field’ mismatch (that condition is being checked in attributes of ’check.fieldunderattack.f), then we have the classifier of ’Field Under Attack’ matched.

Classifier ‘King’ is a minimal classifier, so it has only nuclear classifiers as attributes. Well, if we get an instance with the value of figure type equal to 6 (symbolic presentation of the king), ‘king’ is classified. Here we have an additional condition checking in king.cx and king.cy, which means that the coordinates of king must be equal to the coordinates of a ‘field’ in classifier ‘field under attack’, which will activate matching of ‘check’.

We develop structures similar to those of OOP, including 3 dimensions of a natural language grammar, “have”, which is the relation between composers and attributes in OOP classes, “be”, which is the inheritance of classes from each other and “do”, which is the ability to define methods in classes. Other than HBD dimensions, we aim to provide the ability to define virtual classes, as it is in OOP, e.g., “field under attack” concept of chess is virtual in its essence, so we need to provide the way to do it. We need to start definition of any problem from basic types, this is in parallel to OOP can be compared with the declaration of built-in types.

In comparison with human acquisition, the currently provided models in Solver have some restrictions, particularly the human acquisition can be performed from examples, by inductive learning, while Solver at the moment implements only one of acquisition ways, by certain exact models provided by the expert.

Transferring of expert models is adequate to the interaction of an expert and another person, where the expert starts passing mentals and classifiers from a certain level (background) and the human can acquire different levels and types of mentals, such as virtual concepts similar to OOP abstract classes, simple rules and different composites. The acquisition is performed iteratively, level by level, starting from the level Solver knows, in chess we consider the first level the set of 4 concepts (figure type, color and x/y coordinates). The rest of systemic classifiers acquired from the expert are classifiers that compose HBD relations and new rules.

For example, the acquisition of a check requires an acquisition of chess concepts 'king', 'field under attack', which are acquired similar to the acquisition by a human.

At the moment, we have transferred the main classifiers of chess to Solver, and experiments show that each of them has been adequately acquired by the system.

3. Matching to Systemic Classifiers

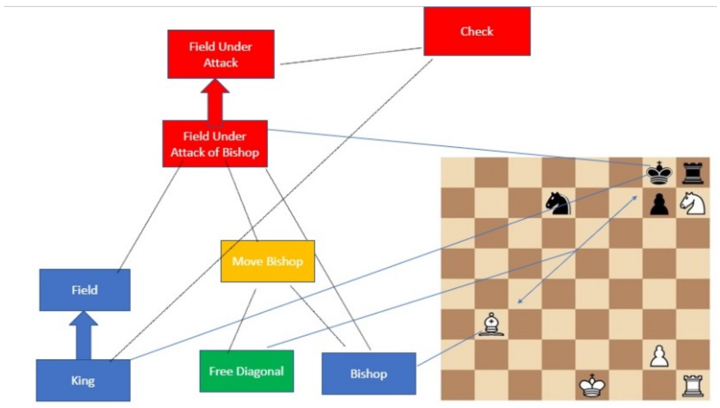


Fig. 4. Chess example with classifier ‘check’ matching.

In this chapter we will demonstrate the matching algorithm on the example of the chess complex concept “mate”.
Checkmate (often shortened to mate) is a game position in chess and other chess-like games, in which a player's king is in check (threatened with capture) and there is no way to remove the threat. Checkmating the opponent wins the game [7].

We define ‘mate’ as a Composite Classifier with the following attributes:

a. ‘Check’, which is a Composite Classifier,
b. ‘King Has no Move’, which is a Dynamic Classifier,
c. ‘King Has no Defence’, which is also a Dynamic Classifier.

Let’s go deeper into the whole matching algorithm for this example.

In Figure 4, an example of chess concept ‘checkmate’ or ‘mate’ is shown. Let's see which Minimal Classifiers are activated by our matching algorithm. The Minimal Classifier ‘White Rook’ activates because its attributes match (figure colour is white; figure type is equal to the figure type of rook).

Then, by the bottom-up movement, it activates its parent ‘Rook’, then ‘Rook’ activates ‘Figure’ and finally ‘Figure’ activates ‘Field’. From the left we see a similar example for the ‘Empty Field’. Here ‘Rook’ and ‘White Rook’ are connected by the relationship *have*. Other figures also activate similarly (‘black king’ -> ‘king’ -> ..., ‘white bishop’ -> ’bishop’).

In Figure 4, Composite Classifier ‘Check’ matching is visualized. ‘Check’ has the following attributes – 1. ‘Field Under Attack’, which is activated by its child ‘Field Under Attack of Bishop’, the latter is activated by its attributes ‘Field’ (Min. Classifier), ‘Move Bishop’ (Action, activates with set ‘Free Diagonal’ and Min. Classifier ‘Bishop’) and ‘Bishop’ (Min. Classifier) and 2. ‘King’, which must be the same ‘King’ that activates ‘Field’, which was mentioned in 1.

For Dynamic Classifier ‘King has no Move’ matching steps are as follows:

1. Precondition - Matching of Minimal Classifier ‘King’ (parent of ‘Black king’).
2. Action – Matching of Action ‘Move King’, which can (and should be applied for matching the Dynamic) be applied to the situation. Actions should be applied the number of times equal to the variable ‘depth’, which is 1 by default.
3. Postcondition – After applying the action(s) (blue arrow) Composite Classifier ‘Check’ should be matched (its matching was described previously) for every applied action (in this case, 3 possible moves for the king, after each ‘Check’ will be matched).

For Dynamic Classifier ‘King has no Defence’ matching steps are as follows:

1. Precondition - Matching of Minimal Classifier ‘King’ (parent of ‘Black king’).

2. Action – Matching of Action 'Move Figure' (in this case, figure should have the same figure colour as king in Precondition), which can (and should be applied for matching the Dynamic) be applied to the situation. Actions should be applied the number of times equal to the variable 'depth', which is 1 by default.
3. Postcondition – After applying the action(s) (blue arrow) Composite Classifier 'Check' should be matched (its matching was described previously) for every applied action (in this case, 3 possible moves for black pawn, one for black rook, and 6 for black knight, after each 'Check' will be matched).

To sum up, we brought detailed visualization for matching Composite Classifier 'Mate'. As in previous figures, blue blocks are for Min. Classifiers, Green for Sets, Yellow for Actions, grey for Dynamics and red for Composites. White arrow is for action applying and colourful arrows are for derivative to parent relation (relation *have*).

Matching and classification algorithms, in comparison with human approaches, appear to be very similar.

First level of matching is transforming input situations from their natural presentation to the numerical or understandable format. This step is currently in progress in Solver and machine learning techniques are being used to transform situation from chess board images to a set of input instances.

Second level of classification and matching is processing the situation and finding the expected instances of classifiers (can be both do classifiers and systemic). This is done already and current experiments show the adequacy of the algorithms to human approaches. For systemic classifiers matching is done level by level, searching from top to bottom, e.g., to classify the situation and see if there is check or no it can search for check iteratively to bottom. First it will look for check itself and then match its components and related classifiers, 'king' and 'field under check' and so until it reaches to already classified instances.

Third level is using effectors to provide the output. Currently Solver simply names the matched classifiers as output and provides its instance in its presentation. As an improvement to it we will transform that output to situations on chess board images.

As a summary to this section, adequacy of matching to systemic and do classifiers was demonstrated, which can be improved by providing sensors and effectors, which will let interact to Solver more naturally.

4. Conclusion

From the urgent cognition modelling and understanding problems we consider the problem of knowledge acquisition and classification of realities by mentals, which extend Object Oriented programming language abilities. We also consider RGT class of combinatorial problems, particularly chess, as a kernel problem. Solver is backed by Systemic classification theory, and acquisition algorithms were implemented, and in the current work we have the following:

1. Knowledge acquisition algorithms and structures developed in Solver 18 are experimented chess. Different chess concepts, including mate were acquired by Solver adequately. In this paper, we have also shown how it is being modelled and how knowledge is being acquired by RGT Solver 18.
2. Matching algorithms for different types of acquired classifiers were experimented for chess and the adequacy was demonstrated, particularly chess concepts were matched to situations, including 'mate'.
3. Provided experiments demonstrate the adequacy of acquisition and classification of situations from input by specific instances Solver expects and it behaves like do classifiers

described in [1, 2] while classifiers can be queried from their attributes thus showing their systemic essence and can be queried from not only expected certain instance based situations but also from natural presentation of realities and situations (it is in next steps of development).

4. Plans for the next stage are the following: A) experimenting the performance of sensors implementation for chess, as for now we pass situations by instances of certain classifiers (i.e. transforming chess board images to RGT presentation); B) Integration and experimenting of planning in chess endgames (e.g., rook vs king).

References

[1] E.Pogossian, "Towards adequate constructive models of mental systems", *Proceedings of International Conference of Computer Science and Information Technologies (CSIT),* pp. 37-46, 2017.
IEEEXplore ttp://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=8307363

[2] E. Pogossian, "On the way to dominating cognition", *Transactions of IIAP NAS RA, Mathematical Problems of Computer Sciences,* vol. 48, pp. 79-91, 2018.

[3] S. Grigoryan, N. Hakobyan and H.Vrtanesyan "Object –oriented modeling of matching to systemic classifiers", *Transactions of IIAP NAS RA Mathematical Problems of Computer Sciences,* vol. 48. pp. 115-121, 2018.

[4] 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,* Salamanca, Spain, p. 16, 2011.

[5] 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.

[6] E. Pogossian, "Adaptation of combinatorial algorithms", Academy of Sci. of Armenia, p.293, 1983, Yerevan

[7] [Online]. Available: https://en.wikipedia.org/wiki/Checkmate

Submitted 22.06.2018, accepted 04.12.2018.

Սիստեմիկ դասակարգիչների յուրացման և դասակարգման փորձարկումներ

Ս. Գրիգորյան և Ն. Հակոբյան

Ամփոփում

Ուսումնասիրվում է յուրացման և դասակարգման կարողությունները մեքենայի համար մոդելավորելու հնարավորությունը: Մեր հետազոտական ուղղվածությունը հիմնված է ալգորիթմների հիմնադիրների գաղափարների վրա, որը թույլ է տալիս մեզ կառուցողական կերպով մոդելավորել հաշվողականությունը՝ այլ մտավոր գործողությունները կառուցողական կերպով մոդելավորելու նպատակով ([1,2]): Ներկայացվում են սիստեմիկ դասակարգիչների համակարգչային յուրացման ՕԿԾ մոդելներ և դրանց ճշտության փորձարկումներ շախմատի համար՝ RGT կոմբինատոր խնդիրների դասի շրջանակներում:

Экспериментирование компьютеризации и проверки системных классификаторов

С. Григорян и Н. Акопян

Аннотация

Исследуется возможность компьютерного моделирования приобретения системных классификаторов и проверки их правильной работы. Исследовательское направление, которую мы придерживаемся, продолжает идеи изобретателей алгоритмов, позволяя конструктивно моделировать вычислимость, с целью конструктивного моделирования других ментальных функций [1,2]. Представлены ООП модели компьютерного приобретения системных классификаторов и эксперименты проверки их правильной работы для шахмат в рамках класса RGT комбинаторных проблем.