ABSTRACT LANGUAGE OF THE MACHINE MIND

Ján KOLLÁR, Milan SPIŠIAK, Michal SIČÁK
Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics
Technical University of Košice, Letná 9, 042 00 Košice, tel. 055/602 2561
E-mail: {jan.kollar,milan.spisiak,michal.sicak}@tuke.sk

ABSTRACT

In human-computer communication, abstract language of the machine mind should be represented by language concepts that reflect language concepts of humans. Language concepts are abstracted in the process of conceptualization and their meaning is given by meaning of symbolized reality of surrounding world. That is why each concept represented internally has meaning by definition, following semiotic approach. In this paper we present the description of our solution to the machine mind, so far restricted to regular languages. To show that the mind is the language not just a grammar, we introduce briefly the principle of symbolization and conceptualization, omitting semantic aspects of thought. As a result, we get a supercombinator form for the internal language of the mind. We do not solve here the language of thought in the sense of machine thinking. Using simple example of story occurring frequently in the animal world, we will simply suppose that the stories represent external structured symbols and are approved be their existence. We are focusing on their internal representation in the machine mind, which has language substance. First, we show how structural explosion in the mind is prevented by well-performed abstraction during conceptualization. Second, introducing formalism of meta-lambda calculus – an application of lambda calculus to regular expression, we present algorithmically evolved abstract internal language of the machine mind in supercombinator form evolved by absorption of structured symbols on the machine input.

Keywords: *Language abstraction, human-computer communication, language mind, meta-semantics, meta-lambda calculus.*

1. INTRODUCTION

Conscious understanding the reality is the inherent property of a human mind. But our hypothesis is that machines could perhaps mimic human language concepts, if a machine alternative for an abstract internal language of humans is evolved.

In this sense, machine understanding is an automated evolution and interpretation of language concepts, sufficient for exchanging semantically similar complex symbols between humans and machines. Externally communicated symbols such as music, pictures, pictograms, words, sentences, paragraphs, books, etc. in informal/natural languages, or statements, programs, models, graphs, etc. in formal languages may have actually a very complex structure, which is approved on input of a human or machine by its existence. Semantic similarity is the matter of the same interpretation by communicating actors. In our opinion, it is necessary to have the detailed structure of dynamically evolvable machine mind before considering semantic aspects of machine thinking.

Human-computer communication is at higher level of abstraction than simple interaction, since information significant for humans is reflected by machines at the higher level of language abstraction, i.e. by concepts that are internal languages for external symbols. The hypothesis is that we are not thinking in natural languages of loud symbols that we recognize in our thoughts but in internal calm concepts that we do not recognize at all. As we will see these internal concepts consist of sub-concepts formed by acquiring complex symbols on input. Hence, we follow Shaymyan's semiotic approach [17] not however restricted to natural languages. As we feel, there are many application areas, in that language abstraction in human-computer and computer-computer communication is interesting using also formal languages – some of them we introduce below.

Without-doubt, the evolution of languages in wider context of human society is activated by communication [16] since humans are symbolic beings. Following symbolic Chomsky's approach [6], we can see that language structures consist of external symbols, that are abstracted to grammars. In reverse direction, automata are derived based on grammars and they recognize language structures. To prevent confusion, grammatical terminal symbols used in Chomsky hierarchy of languages are not externally communicated symbols. They are fine-grained abstractions for lexical units understandable to humans.

The crucial point in an efficient communication is language abstraction [3,5,12]. The need for abstraction can be found in many application areas, such as information systems, domain specific languages, telecommunication systems, computer networks, etc. Generating domain specific languages starting with abstract syntax [2,15] are examples of solutions based on abstraction.

From the viewpoint of strengthening communication between humans and computers, the measure of automation should be significantly increased. Machines must derive language concepts in an automated manner when acquiring information on their input. Grammatical inference [10] comes out from Gold's theorem [7] which states that it is impossible to identify any of the four classes of languages in the Chomsky hierarchy in the limit using only positive samples. On the other hand, the machine is a passive actor in this approach, not able to formulate questions. As we feel, the ability to formulate questions by machines is a very important property, which is supported by the ability to reconstruct external symbols from internal concepts in our solution.

The complexity of conceptualization is usually solved using probabilistic approaches, see [4,10,11]. In our experiment [8] with genetic evolution of programs from context-free (CF) grammars, genetic string is used in a deterministic manner to generate base level programs from extended

Backus Naur form metalevel.

Using crossovers in genetic evolution [13] is based on the idea that material processes of genesis and growth of living organisms are counterparts of immaterial grammatical processes. Our approach is different. In our opinion, systems are deterministic, just they are unknown to us in detail, as stated by Peterson [14]. The finest level, which we aim to recognize deterministically are not the biochemical processes in human brain, "just" immaterial processes in forming the language of the human mind. Considering determinism, a problem of structural explosion arises, which is solved by Smith at automata level adding the auxiliary variables, see [18]. Our approach is different – we are focusing to language and metalanguage level with automated derivation of the automaton.

There is a strong need for the distributed structures for deterministic finite state automata (DFA), along with the ability for their matching in an associative memory [19]. This follows us to think about a fine grained parallel representation of the machine mind language. In this paper, we present the solution in the supercombinator form of abstract language of machine mind, which has been derived algorithmically. We also discuss the results and promissing properties for further research.

But first, in Section 2, we introduce the essence of semiotic binding of symbolized reality to language concepts, since it is crucial for understanding of mutual and nonseparable binding of syntax and semantics in our approach: grammars are structural abstractions of languages, but at the same time, grammars and languages are synonyms. Abstract language of the machine mind, presented in this paper, belongs to the category of regular languages in Chomsky hierarchy.

2. SYMBOLIZATION AND CONCEPTUALIZATION OF REALITY BY MACHINES

Each symbol on input is symbolized and conceptualized, before it is represented in an internal form of abstract language of the mind. We introduce the principles of symbolization and conceptualization in the following subsections.

2.1. Principle of Symbolization

As an illustrative example, let us introduce the reality taken from an animal world: the dog and the cat are animals and they are usually enemies. Both can be visually observed by human beings, and their pictures can be recorded and reproduced by machines. This story is depicted in Fig. 1.

In general, reality is represented in a very different ways, for example by photographs, by pictures, by sounds, by words, etc. For the purpose of human-computer communication, communicated symbols must be in the form understandable to a human being. Let a picture of the dog is designated by uppercase D and the picture of the cat by C.

When C or D should be processed by machine in the way analogical to humans, they must by symbolized formally. Symbolization is the process of forming the bidirectional associations of formal grammatical symbols to recorded representations of real objects communicated as external symbols. The process of symbolization is highly dependent on aspects that are symbolized. For the purpose of simplicity we suppose that transduction [1] is integrated part of symbolization. Depending on aspects analysed and transduced, original is less or more reproducible. For our purposes, we use just a very coarse symbolization, in which complex externally communicated symbol – a picture of the dog differs from a picture of the cat, using different grammatical symbols. As a result, we have an ability to reproduce external symbols on output, what is the most important here.

By symbolization of the dog and the cat, we obtain the set of associations (1)

$$\{\, d \leftrightarrow D,\ c \leftrightarrow C \,\} \tag{1}$$

where formal (grammatical) symbol d symbolizes recorded picture D of the realistic dog, and symbol c symbolizes recorded picture C of the cat.

In this way, whenever machine recognizes a picture of the dog D on its input, it replaces this picture by symbol d for internal processing, and vice versa, instead of d, it produces picture D on its output.

At this stage of recognizing reality by machines, we can also notice the difference between grammar and the language: if association set (1) does not exist, symbols d and c are grammatical symbols, since they have no meaning. Otherwise, d is formally bound to an informal meaning of a dog, and c is bound to an informal meaning of a cat. Hence, d and c with associated meaning are primitive languages that represent facts. Clearly, if machine displays picture D instead of symbol d, it is understandable to a human being.

The communication is based on exchanging not just facts but also complex structured symbols, such as user interfaces, programs, models, stories, songs, etc. To illustrate the principle of conceptualization, let us consider again the structured symbol – the story of the dog chasing the cat in Fig. 1. Looking at the story from the left to the right, the sequence of picture records (2) is observed

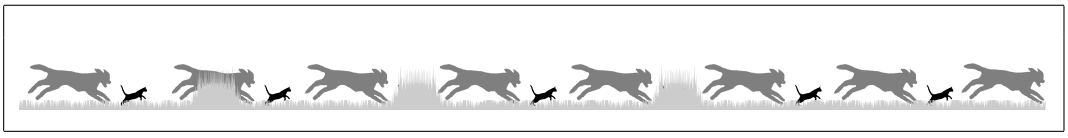

Fig. 1 Structured reality: the dog chasing the cat

$$D\ C\ D\ C\ D\ D\ C\ D\ D\ C\ D\ C\ D \tag{2}$$

which expresses depicted story, and based on association set (1) it is symbolized to the form (3)

$$d\ c\ d\ c\ d\ d\ c\ d\ d\ c\ d\ c\ d \tag{3}$$

Form (3) is the simplest (i.e. string) form of the language concept. Considering associated meaning it is the concrete regular language. Ignoring associated meaning, it is grammatical string, belonging to the category of regular grammars (regular expressions) in Chomsky hierarchy.

As we will see in subsection 2.2, abstraction of conceptualization is more powerful than that of symbolization.

2.2. Principle of Conceptualization

Concepts are internal representations of external symbols in the form of languages. Concepts may be very primitive, but also very complex. Conceptualization is crucial in language evolution, since it is the process of evolving (and changing) the language during communication.

Considering the story in Fig. 1 in the form (3), it can be expressed more abstractly: for example, the dog does not start to chase the cat until a cat appears, so prefix $d\ c$ is the proposition to start chasing, and chasing is finished when the cat escapes, hence story terminates by d. Chasing itself consists of repeated running and jumping of animals, but the cat sometimes disappears behind the tall grass. Omitting the reasons, we introduce the abstraction as a possible alternative, to be able to show that more abstract representation of concepts prevents structural explosion.

Then more abstract concept of the story can be expressed by regular language in the form of regular expression (4)

$$d\ c\ \{\ d\ [\ c\ ]\ \}\ d \tag{4}$$

where transitive closure $\{\ r\ \} = \varepsilon \mid r \mid r\ r \mid r\ r\ r \mid \ldots$ expresses repeated (and possibly empty) sequence of regular expression r (in our case $r = (d\ [c])$, and optional occurrence $[\ r\ ] = \varepsilon \mid r$ expresses the fact that r (in our case $r = c$) occurs optionally.

The effect of abstraction by conceptualization is such that different number of occurrences of the dog and the cat is irrelevant for the sufficient recognition of acquired story on input. Evidently, conceptualization can increase the abstraction of concepts rapidly, since (5) holds.

$$d\ c\ d\ c\ d\ d\ c\ d\ d\ c\ d\ c\ d \subset d\ c\ \{\ d\ [\ c\ ]\ \}\ d \tag{5}$$

It is also noticeable, that communicated symbols may be matched by both partners in communication successfully, even if their internal concepts are different or wrong, because they may be corrected in later evolution during communication. This perfectly corresponds to the fact that communication can be successful even if current internal representation of concepts in the mind is different.

For example, if (more abstract) language concept known to a human is (4) and (less abstract) language concept known to a machine is (3) then just less abstract concept (3) is usable in communication.

The wrong concept known to machine $d\ c\ d\ \{\ d \mid c\ \}\ d$ is correctly usable, until two subsequent $c\ c$ does not appear in communication.

If both partners recognize the same high abstract language concepts (4), the communication is more flexible. For example, both will agree that even $d\ c\ d$ is the same story (such that the cat appears and escapes after the first dog's jump).

Let us present in Section 3 the problem of structural explosion, which yields us our solution to the representation of abstract internal language, introduced in Section 4.

3. STRUCTURAL EXPLOSION

When recognizing a sentence (a simple kind of regular language concept) by a deterministic finite-state automata, we are able to do so by abstract interpretation, see [9], which is a meta-execution, i.e. execution of a meta-language. When regular language is meta-executed, deterministic finite-state automaton is derived. But then the problem of structural explosion arises. On the other hand, the solution to this problem yields surprisingly simple representation of the abstract language of the machine mind in supercombinator form.

Let us explain first what the structural explosion problem means in the context of abstract language of the mind.

Using `SUM` for n-nary sum $(|\ldots|)$ metaoperation, `SEQ` for binary right associative sequencing metaoperation, `TERM` for terminal symbol, `EXP` for unary $((\,))$ metaoperation of subexpression, `OPT` for unary option $([])$, and `CLS` for unary transitive closure $(\{\})$, regular expression $d\ c\ \{\ d\ [\ c\ ]\ \}\ d$ transformed to the syntactic tree and the equivalent graph form is depicted in Fig. 2.

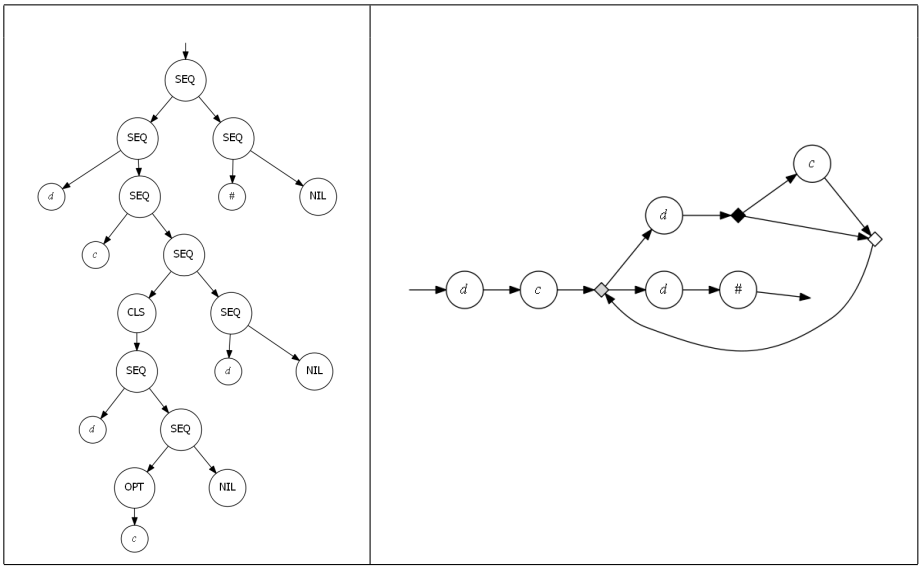

Fig. 2 Syntactic tree and graph of $dc\{d[c]\}d$.

Abstract interpretation on the syntactic tree just sequentially emulates parallel flow of marks in regular expression graph, but functionally both forms are equivalent. Derived DFA is introduced in Fig. 3.

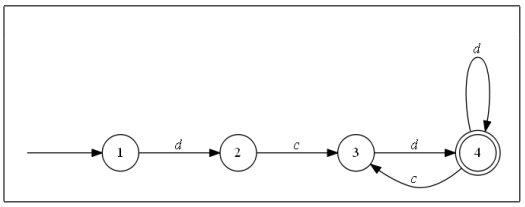

Fig. 3 DFA for $dc\{d[c]\}d$.

At this point, the problem of structural explosion can be explained considering our animal story. Suppose that except the dog and the cat, the mouse M (in symbolic form m) appears as the third animal. Let us consider two other stories being acquired as structured symbols on machine input: (1) the cat chasing the mouse, and (2) the dog chasing the mouse. Acquiring multiple structured symbols (regular languages) subsequently is equivalent to their summarization. Summarizing three structured symbols, we get regular expression and derived DFA according Fig. 4.

Clearly, the structure of regular language and corresponding DFA grows rapidly. Summarizing hypothetically thousand symbols acquired on input, the effect of structural explosion is evident.

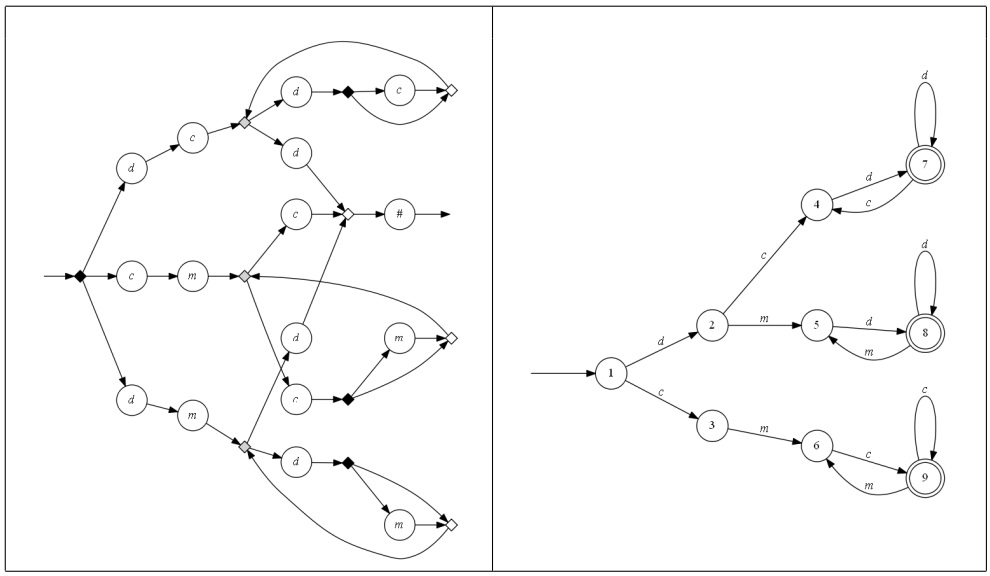

Fig. 4 Graph of $dc\{d[c]\}d|cm\{c[m]\}c|dm\{d[m]\}d$ and derived DFA.

As can be seen, many similarities occur in explosively evolved language and derived automaton. This tends us to decompose the language (not the automaton) in the way as a child dismounting the toy to see all its parts. But in contrast to a child, which is not able to compose the toy if missing some part, the machine mind minimizes the set of languages in such way that original language and reality is reproducible.

As a result we get abstract language as a minimum set of abstract languages in supercombinator form, that are composable dynamically, i.e. by the application, even in parallel. The essence of method which we used when evolving abstract language of the machine mind algorithmically, we present in Section 4.

4. ABSTRACT LANGUAGE EVOLUTION

As we will see in subsection 4.2, abstract language is a minimal set of abstract languages, derived from acquired structured symbol on input. Abstract language itself has no meaning, until it is applied to approved elementary symbols, representing facts. Sets of elementary symbols are derived during abstract language evolution.

The method of evolution uses meta-lambda calculus, which we propose in the form of the application of lambda calculus to regular expressions.

4.1. Meta-lambda Calculus

Lambda expression e is defined by the rule (6)

$$e \to a \mid x \mid e_1\ e_2 \mid \lambda x.\ e \tag{6}$$

where a is a constant, x is a (lambda) variable, $e_1\ e_2$ is the application of an expression e_1 to an expression e_2, and $\lambda x.\ e$ is the lambda abstraction, which designate a function with lambda variable x (formal parameter) and e is the expression which defines the value of lambda abstraction application.

A regular expression r is defined by rule (7).

$$r \to a \mid r_1 + \ldots + r_n \mid r_1 | \ldots | r_n \mid (r) \mid [r] \mid \{r\} \tag{7}$$

where a designates a terminal, $r_1 + \ldots + r_n$ is the sequence of regular expressions, $r_1 | \ldots | r_n$ is the sum of regular expressions, and parentheses, brackets and curly brackets have this meaning: $(r) = r$, $[r] = \varepsilon | r$, and transitive closure $\{r\} = \varepsilon | r | rr | rrr | \ldots$.

Now, we are ready to express meta-expression r in meta-lambda calculus form (8)

$$\begin{array}{rcl} r & \to & (\lambda x.\ x)\ a \\ & \mid & (\lambda x_1.\ \ldots\ \lambda x_m.\ r_1\ + \ldots +\ r_n)\ a_1\ \ldots\ a_m \\ & \mid & (\lambda x_1.\ \ldots\ \lambda x_m.\ r_1 \mid \ldots \mid r_n)\ a_1\ \ldots\ a_m \\ & \mid & (\lambda x_1.\ \ldots\ \lambda x_m.\ r)\ a_1\ \ldots\ a_m \\ & \mid & (\lambda x_1.\ \ldots\ \lambda x_m.\ (r))\ a_1\ \ldots\ a_m \\ & \mid & (\lambda x_1.\ \ldots\ \lambda x_m.\ [r])\ a_1\ \ldots\ a_m \\ & \mid & (\lambda x_1.\ \ldots\ \lambda x_m.\ \{r\})\ a_1\ \ldots\ a_m \end{array} \tag{8}$$

in which all lambda abstractions are supercombinators (lambda abstractions that have no free variables) and sequence operation in regular expression is designated by $(+)$ to distinct it from application designated by space.

4.2. Abstract Language of the Machine Mind Evolution and Results

Decomposing regular expression to maximum number of supercombinators in the first step, we get the set of supercombinators (occurring multiply) and all arguments for each of them. For example, the decomposition of regular expression $dc\{d[c]\}d$ introduced in Table 1 yields 13 supercombinators. L^{12} is the root, since its application reproduces structured symbol abstracted to $dc\{d[c]\}d$. It may be proved that $(L^{12}\ d\ c = dc\{d[c]\}d)$ holds.

Table 1 Supercombinators and their arguments for decomposed regular expression $dc\{d[c]\}d$

Supercombinators			Arguments
L^0	=	$\lambda x_1.\ x_1$	d
L^1	=	$\lambda x_1.\ L^0\ x_1$	d
L^2	=	$\lambda x_1.\ x_1$	c
L^3	=	$\lambda x_1.\ \lambda x_2.\ L^1\ x_1\ +\ L^2\ x_2$	$d\ c$
L^4	=	$\lambda x_1.\ x_1$	d
L^5	=	$\lambda x_1.\ L^4\ x_1$	d
L^6	=	$\lambda x_1.\ x_1$	c
L^7	=	$\lambda x_1.\ [L^6\ x_1]$	c
L^8	=	$\lambda x_1.\ \lambda x_2.\ L^5\ x_1\ +\ L^7\ x_2$	$d\ c$
L^9	=	$\lambda x_1.\ \lambda x_2.\ \{\ L^8\ x_1\ x_2\ \}$	$d\ c$
L^{10}	=	$\lambda x_1.\ \lambda x_2.\ L^3\ x_1\ x_2\ +\ L^9\ x_1\ x_2$	$d\ c$
L^{11}	=	$\lambda x_1.\ x_1$	d
L^{12}	=	$\lambda x_1.\ \lambda x_2.\ L^{10}\ x_1\ x_2\ +\ L^{11}\ x_1$	$d\ c$

Considering regular expression (9)

$$dc\{d[c]\}d|cm\{c[m]\}c|dm\{d[m]\}d \tag{9}$$

we would obtain 40 supercombinators ($L^0 \ldots L^{39}$), root being represented according (10)

$$L^{39} = \lambda x_1.\ \lambda x_2.\ \lambda x_3.\ L^{12}\ x_1\ x_2 \mid L^{25}\ x_2\ x_3 \mid L^{38}\ x_1\ x_3 \tag{10}$$

such that $L^{39}\ d\ c\ m = dc\{d[c]\}d|cm\{c[m]\}c|dm\{d[m]\}d$.

In the second step, multiple occurrences of supercombinators are removed considering equality of two abstract languages.

Languages L^1 and L^2 are equal ($L^1 = L^2$) if they are of the same arity and they are defined by equal expressions. Expressions e^1 and e^2 are equal ($e^1 = e^2$), depending on

their forms, see (11).

$$
\begin{array}{rcl}
x^1 & = & x^2, \\
 & & \text{if } x^1 = x \wedge x^2 = x \\
L^1\ x_1^1\ \ldots\ x_n^1 & = & L^2\ x_1^2\ \ldots\ x_n^2, \\
 & & \text{if } L^1 = L^2 \wedge x_1^1 = x_1^2 \wedge \ldots \wedge x_n^1 = x_n^2 \\
[\ e^1\ ] & = & [\ e^2\ ], \\
 & & \text{if } e^1 = e^2 \\
\{\ e^1\ \} & = & \{\ e^2\ \}, \\
 & & \text{if } e^1 = e^2 \\
e_x^1 + e_y^1 & = & e_x^2 + e_y^2, \\
 & & \text{if } e_x^1 = e_x^2 \wedge e_y^1 = e_y^2 \\
e_1^1 \mid \ldots \mid e_n^1 & = & e_1^2 \mid \ldots \mid e_n^2, \\
 & & \text{if } e_1^1 = e_1^2 \wedge\ \ldots\ \wedge e_n^1 = e_n^2
\end{array}
\tag{11}
$$

The result of compression for regular expression $dc\{d[c]\}d$ yields total minimum number of 8 supercombinators for internal abstract language, see Table 2, where root is L^7 ($L^7\ d\ c = dc\{d[c]\}d$), and compression ratio is $13/8 = 1.62$.

The result of compression for regular expression $dc\{d[c]\}d|cm\{c[m]\}c|dm\{d[m]\}d$ yields total minimum number of 9 supercombinators for internal abstract language, see Table 3, where root is L^8 ($L^8\ d\ c\ m = dc\{d[c]\}d|cm\{c[m]\}c|dm\{d[m]\}d$), and compression ratio is $40/9 = 4.444$.

Table 2 Abstract language for symbol $dc\{d[c]\}d$

Supercombinators			Arguments
L^0	=	$\lambda x_1.\ x_1$	$\{\ d,\ c\ \}$
L^1	=	$\lambda x_1.\ L^0\ x_1$	$\{\ d\ \}$
L^2	=	$\lambda x_1.\ \lambda x_2.\ L^1\ x_1 + L^0\ x_2$	$\{\ d\ c\ \}$
L^3	=	$\lambda x_1.\ [L^0\ x_1]$	$\{\ c\ \}$
L^4	=	$\lambda x_1.\ \lambda x_2.\ L^1\ x_1 + L^3\ x_2$	$\{\ d\ c\ \}$
L^5	=	$\lambda x_1.\ \lambda x_2.\ \{L^4\ x_1\ x_2\ \}$	$\{\ d\ c\ \}$
L^6	=	$\lambda x_1.\ \lambda x_2.\ L^2\ x_1\ x_2 + L^5\ x_1\ x_2$	$\{\ d\ c\ \}$
L^7	=	$\lambda x_1.\ \lambda x_2.\ L^6\ x_1\ x_2 + L^0\ x_1$	$\{\ d\ c\ \}$

In general, if m is a minimum number of supercombinators L_k, $k = 1 \ldots m$, and $Args_k$ is the set of arguments for language L_k, then the number of all arguments $\sum_{k=1}^{m} \mid Args_k \mid$ selected during internal abstract language derivation is equal to the number of possible applications, that potentially can reproduce all symbols acquired by structured symbol, which is reproducible by application of the root supercombinator.

Table 3 Abstract language for symbol $dc\{d[c]\}d|cm\{c[m]\}c|dm\{d[m]\}d$

Supercombinators			Arguments
L^0	=	$\lambda x_1.\ x_1$	$\{\ d,\ c,\ m\ \}$
L^1	=	$\lambda x_1.\ L^0\ x_1$	$\{\ d,\ c\ \}$
L^2	=	$\lambda x_1.\ \lambda x_2.\ L^1\ x_1 + L^0\ x_2$	$\{\ d\ c\ \}$
L^3	=	$\lambda x_1.\ [L^0\ x_1]$	$\{\ c,\ m\ \}$
L^4	=	$\lambda x_1.\ \lambda x_2.\ L^1\ x_1 + L^3\ x_2$	$\{\ d\ c,\ c\ m,\ d\ m\ \}$
L^5	=	$\lambda x_1.\ \lambda x_2.\ \{L^4\ x_1\ x_2\ \}$	$\{\ d\ c,\ c\ m,\ d\ m\ \}$
L^6	=	$\lambda x_1.\ \lambda x_2.\ L^2\ x_1\ x_2 + L^5\ x_1\ x_2$	$\{\ d\ c,\ c\ m,\ d\ m\ \}$
L^7	=	$\lambda x_1.\ \lambda x_2.\ L^6\ x_1\ x_2 + L^0\ x_1$	$\{\ d\ c,\ c\ m,\ d\ m\ \}$
L^8	=	$\lambda x_1.\ \lambda x_2.\ \lambda x_3.\ L^7\ x_1\ x_2 \mid L^7\ x_2\ x_3 \mid L^7\ x_1\ x_3$	$\{\ d\ c\ m\ \}$

5. DISCUSSION

We were concentrated in this paper to the abstract (internal) language of the machine mind, obtaining the following results.

1. There is no danger of structural explosion anymore, since all terminals originally positioned in syntactic tree leaves are now represented by one identity supercombinator application, see Table 2 and/or Table 3.

2. Similar symbols on input exploit already stored elementary supercombinators – two new stories yield increasing the number of supercombinators by one, compare Table 2 and Table 3.

3. Abstract language is non-redundant, i.e. multiple occurrences of supercombinators do not exist, although they occur in a hidden form in a source regular expression, see decomposed form in Table 1.

4. Upper indices of supercombinators can be understood as positions in a highly parallel architecture, and supercombinators can be fetched by matching before they are applied. Moreover, the application of a supercombinator to arguments can be potentially implemented by sending the values of arguments to supercombinator inputs in a highly parallel dataflow manner.

5. Considering supercombinators bodies, they are in the form of regular expressions again, but they contain the applications of other supercombinators instead of terminal symbols. It yields intensive inner communication and clearly, a new kind of machine mind ar-

chitectures is highly required.

6. Evolving the abstract language from an input structured concept, which is approved by default, many other subconcepts are implicitly derived and approved. It means, that the communication based on complex structured symbols evolves internal language more rapidly, than if using just simple symbols.

Our approach comes out from symbolism and conceptualism. The goal of our research is to find universal meta-computational algorithm to the evolution of the machine mind, considering both formal and informal languages. The algorithm should produce the language structure of the machine mind, which should be appropriate for formal reasoning by an analogy to human thinking. In this paper we present the solution restricted to the category of regular languages. Although our approach is different than that in cognitive science based on connectionist models suported by cognitive semantics, there are some common points.

First, the internal language of the mind in our solution is represented by the meta-computational algorithm which computes the structure similar to neural networks. Structural analogies between elementary supercombinators and neurons, as well as between activating applications and synapses are evident. The answer to the question if there exists some mapping between supercombinator form and neural networks is not so simple.

Second, meta-computational process of the evolution of the machine mind dynamically varies the internal structure of the mind, which is promising property taking into account that languages are changing during communication permanently.

The main contribution for our future research is the fact, that minimum supercombinator structure of the mind is a possible criterion for well-performed automated abstraction during conceptualization.

6. CONCLUSION

We present three levels in the automated language evolution. The outer level of symbolization of reality is an analogy to human perception. The middle level of conceptualization evolves structured symbols – regular languages based on approval during communication. The inner level contains abstract language of the machine mind derived in the form of minimum set of supercombinators and their arguments – sequences of symbols.

The essential impulse for doing this work is simple consideration, that human beings are thinking and evolving its languages when communicating with the external world and/or with their internal minds. By an analogy to this process, we solve the problem of the representation of abstract internal language of machine mind algorithmically. As a result of automated evolution, highly parallel and structurally non-explosive abstract internal language is derived, which reflects reality dynamically, i.e. by applications, not by data.

As we feel, it is promissing step and the proposition for reasoning about the language concepts in the machine mind.

ACKNOWLEDGEMENT

This work was supported by project VEGA 1/0341/13 “Principles and methods of automated abstraction of computer languages and software development based on the semantic enrichment caused by communication”.

REFERENCES

[1] BARSALOU, L.W.: Perceptual symbol systems, Behavioral and brain science,22, 1999, 577-660.

[2] BAČÍKOVÁ, M. – PORUBÄN, J. – LAKATOŠ D.: Defining Domain Language of Graphical User Interfaces. In: Slate 2013 : 2nd Symposium on Languages, Applications and Technologies : June 20-21, 2013, Porto, Portugal. - Wadern : Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013, 187–202.

[3] BĚHÁLEK, M. – ŠALOUN, P.: Usage of embedded process functional language as a modeling tool for embedded systems development. In: International Conference on Intelligent Systems, Modelling and Simulation (ISMS) : Jan 2013, IEEE, 2013, 116–121.

[4] CARVALHO, N. R. – ALMEIDA, J.J. – PEREIRA, M.J. – HENRIQUES P. R.: Probabilistic synset based concept location, In SLATE12 Symposium on Languages, June 21-22, Braga, Portugal, 2012, 239–253.

[5] CHODAREV, S.: Development of Domain-Specific Languages Based on Generic Syntax and Functional Composition. Information Sciences and Technologies Bulletin of ACM Slovakia, 2012, FIIT STU, 47–53.

[6] CHOMSKY, N.: Syntactic Structures, 1957. Walter De Gruyter: Mouton classic, ISBN 3-1101-7279-8

[7] GOLD, M.E.: Language Identification in the Limit, Information and Control, 10, 1967, 447–474.

[8] KOLLÁR, J. – PIETRIKOVÁ, E.: Genetic evolution of programs. Central European Journal of Computer Science, September 2014, 2014, 160–170.

[9] KOLLÁR, J.: Formal Processing of Informal Meaning by Abstract Interpretation. Smart Digital Futures 2014, June 18–20, 2014, Chania, Greece. - Amsterdam : IOS Press, 2014, 122–131.

[10] JAVED, F. – MERNIK, M. – BRYANT, B.R. – SPRAGUE, A.: An unsupervised incremental learning algorithm for domain-specific language development, Applied Artificial Intelligence, 22, 7-8, 2008, 707–729.

[11] McKAY, R.I. – HOAI, N.X. – WHIGHAM, P.A. – SHAN, Y. – O’NEILL, M.: Grammar-based Genetic Programming: A survey. Journal of Genetic Programming and Evolvable Machines, 11, 3–4, 2010, Springer US, 365–396, ISSN 1389-2576.

[12] OBRENOVIĆ, N. – POPOVIĆ, A. – ALEKSIĆ, S. – LUKOVIĆ, I.:Transformations of Check Constraint PIM Specifications Computing and Informatics, 31, 5, 2012, 1045–1079.

[13] O'NEILL, M. – RYAN, C. – KEIZER, M. – CATTOLICO, M.: Crossover in Grammatical Evolution. Genetic Programming and Evolvable Machines, 4, 2003, 67–93.

[14] PETERSON, J.L.: Petri Nets: Theory and Modelling the Systems. Prentice–Hall, Inc. Englewood Cliffs, 1981

[15] PORUBÄN, J. – FORGÁČ, M. – SABO, M. – BĚHÁLEK, M.: Annotation based parser generator. In: Computer Science and Information Systems : Special Issue on Advances in Languages, Related Technologies and Applications, 7, 2, 2010, 291–307.

[16] RENFREW, C.: Prehistory: The Making of the Human Mind. Modern Library Chronicles, 2009, 240 pp.

[17] SHAUMYAN, S.: A Semiotic Theory of Language. Bloomington: Indiana University Press, 1987. 352 pp.

[18] SMITH, R. – ESTAN, C. – Jha Somesh – Kong Shijin: Deflating the big bang: fast and scalable deep packet inspection with extended finite automata, ACM SIGCOMM Computer Communication Review, 38, 4, 2008, 207–218.

[19] YANG, Yi-Hua E. – PRASANNA, V.K.: Space-time tradeoff in regular expression matching with semi-deterministic finite automata, INFOCOM, Proceedings IEEE, 10-15 April 2011, 1853–1861.

Received July 7, 2015, accepted January 18, 2016

BIOGRAPHIES

Ján Kollár is Full Professor of Informatics at Department of Computers and Informatics, Technical university of Košice, Slovakia. He received his M.Sc. summa cum laude in 1978 and his Ph.D. in Computer Science in 1991. In 1978-1981 he was with the Institute of Electrical Machines in Košice. In 1982-1991 he was with Institute of Computer Science at the P.J. Šafárik University in Košice. Since 1992 he is with the Department of Computer and Informatics at the Technical University of Košice. In 1985 he spent 3 months in the Joint Institute of Nuclear Research in Dubna, USSR. In 1990 he spent 2 months at the Department of Computer Science at Reading University, UK. He was involved in research projects dealing with real-time systems, the design of microprogramming languages, image processing and remote sensing, dataflow systems, implementation of programming languages, and high performance computing. He is the author of process functional programming paradigm. Currently his research area covers formal languages and automata, programming paradigms, implementation of programming languages, functional programming, and adaptive software and language evolution.

Milan Spišiak graduated (M.Sc.) at the Department of Computers and Informatics of the Faculty of Electrical Engineering and Informatics at Technical University in Košice and currently he is working towards his PhD at the same university. His current research focus on formal and informal languages symbolization and conceptualization.

Michal Sičák is computer science PhD student at the Department of Computers and Informatics of the Faculty of Electrical Engineering and Informatics at Technical University in Košice. He received his M.Sc. summa cum laude in 2014. The subject of his research involves evolution and conceptualization of grammars and languages.