Computational Ivariants in Applicative Model of Object Interaction

Viacheslav Ernstovich Wolfengagen

National Research Nuclear University MEPhI (Moscow Engineering Physics Institute),
Kashirskoye Shosse, 31, Moscow, 115409 Russian Federation
jir.vew@gmail.com

Abstract: This paper is aimed to modeling the object interaction using the applicative computational technology. In this case, starting with a few of generic objects it is possible to study the computational behavior of all the variety of derived objects. It is shown that the computational properties of generic objects when interacting with environment stay unchangeable. This gives rise to a system of free and natural computations based on the ideas of object interaction. The approach in use is significant for data analysis, object recognition and logical forms of determining the object properties in critical information technologies. This research was accomplished at the Institute for Contemporary Education "JurInfoR-MSU" in collaboration with NRNU MEPhI and MIPT under the Project LAMBDA.
[Wolfengagen VE. **Computational Ivariants in Applicative Model of Object Interaction.** *Life Sci J* 2014;11(09s):453-457]. (ISSN:1097-8135). http://www.lifesciencesite.com. 92

Keywords: Applicative computational technology, object interaction, Big Data, computational invariants

1. Introduction

Computing and its development puts a lot of questions, on the most part from which answers either are incomplete, or unknown. At present an exhaustive explanation of the properties of interacting objects as information processes is not yet reached. Direct transfer of the existing in science models of interaction from the field of physics to the area of information objects is impossible. Implementation barriers are largely due to the mathematical apparatus used, which is familiar to physicists, but it is unusual in computer science, not to mention programmers. In this regard, significant result may be the development of a kind of "mathematical adapter" between the two areas.

In this paper the information technology for natural computations was introduced and studied with the limitation to the applicative computational system. Case study for the assumptions of object with object and object with environment interaction was evaluated. The first case gives rise to computations in applicative prestructure while second case leading to computations in applicative structure. The established evaluation map helps to verify and discover the computational invariants.

This is important in an area of Big Data lacking the direct methods and technologies for the purposes of data analysis, object recognition and logical forms of determining the object properties in critical information technologies.

2. Objects and environment

2.1. Fixing the object.

Attempts to fix object, assume it as something "ordinary" (and what is meant by "ordinary", it remains quite unclear and non-augmenting) did not bring the expected result. And it is not negligence of researchers -- they exhibit complete thoroughness and accuracy, and technologists made software, which itself is sound.

To understand the phenomenology of the object the explorer needs more perfect "instrument base", and in this case, faced with constraints of "ordinary" meta-mathematics, which rather awkwardly reflects real object in its notations.

Object also demonstrates its natural diversity: it is both prescription, and the result at the same time. As well can we cope with this?

2.2. Object-as-process.

If we try to imagine object vividly, it turns out that we cannot say definitely -- quite determined, -- what is it. At best we can talk about its *possible presentations*. These possible presentations can be computationally observed, and their relationship forms the framework characterized by special mathematical properties.

Importantly, the object can show itself in an infinite many ways: it has a variety of forms.

"Computational distance" between the individual forms can be described in terms of "redex-contract" ("redex" means "reducible expression"; contract is understood as a limitation, convolution). On the way to the result a reduction of the object is done in the direction from the redex to contract.

During the implementation of the reduction the object interacts with the environment, "captures" disposed therein other objects that are in its sphere of reach, and these objects are recombined in applicative structure:

$$\| \bar{c} \| \, env = c$$

In contrast, the way from the result, an object expansion occurs in the direction from object contract to its redex. During the expansion, the object is synthesized as a *combinator*, releasing previously captured objects which are returned to the environment, while remaining within its reach and scope.

Thus, the object is in a state of transition as redex-contract and is a process.

3. Way of thinking the objects

3.1 Traditional way of thinking.

Entrenched way of thinking is to transfer the abstract and general mathematical theory on some given specific problem domain. Tacitly assumed that all mathematical concepts and the way they interact are well comprehended, and all explanatory system has great expressive power. So large that the chances are good for embedding substantive applied theory into pure theory, to obtain new results. All this led to artificial systems of explanation.

3.2. Artificial explanation.

In developing *object theories* in computer science, which is performed by such a way, the researchers stalked a trap. In fact, the objects once thought of as abstract entities, and strongly linked to the models and explanatory systems prevailing in the discipline of *data structures*.

This resulted in that the artificial computing began to be explained even with more abstraction and artificiality. Trap for the researchers was that the theory of objects were appearing, but were not productive because these theories overgrown cumbersome notations and a lot of complications preventing the perception of details, hence they have not received the development and application, but for these reasons become *intuitively rejected*.

Interest in artificial computing quickly became lost, and the attractiveness of natural computing, by contrast, started to grow rapidly. Return to the "natural" explanatory systems replaces the culture of formalisms and logical forms of reasoning.

3.3. Natural explanation.

An idea to explain the objects and their behavior *naturally* requires the adoption of certain principles. Then, based on these principles, the *explanatory system* is developing. We have to select the central beliefs that will fully characterize the range of the effects, in this case the are computational ones. So, there is one entity -- object, -- and another entity -- environment.

Object interacts with the environment so that the result of evaluation is *placed* in the environment. On the other hand an object in the interaction with the environment receives from environment the values and/or parameters.

4. Computational environment

4.1. Environment representation.

The construction of environment should cause quite comfortable feeling: this is the place where the values of the objects are stored.

Hypothetically, the environment is a universe where there are both "deep" and "peripheral" parts. The details of the deep part are timely unknown, but its structure can help generation of a reasonable assumption. Peripheral part, on the contrary, is good "seen" and is known for not only its structure, but also for all of its components. Talking about from where the environment was taken, it could be argued that "at first it was nothing", but intuition suggests ideas as to imagine this idea of nothing.

The empty environment usually is denoted as a pair of parentheses: ( ).

4.2. Interaction of an object and environment.

4.2.1. Environment. A thesis, that interaction of objects needs the intermediary -- *environment*, is perceived as obviously. At least, currently it does not attract doubts. More rigorously, to initialize an interaction of objects, the structure is needed where they are localized.

Opposite case -- when some "wandering" objects "meet"' other wandering objects, -- is interesting, but this discussion will be postponed for a while. The area of programming gives a case when objects, by some way or otherwise, are already packed by in the environment. Thus a central concept under development is namely the environment which is understood as an environment for computations. Environment is equipped with the programming system, but not wise versa.

Other circumstance is that an object interacts not with all environment at once, but with its partition -- that which will appear "in an area of action" of the object.

4.2.2. Prestructure. An applicative prestructure is used for packing objects. Two aspects of an object -- redex (reducible expression) and the contract, -- reveal in it. In other words, the prestructure gives a representation of computation both in terms of a reduction -- transition from redex to the contract, -- and in terms of expansion -- transition from the contract to redex.

Table 1. Invariance of **I** transformation.

$\| \mathbf{I} \| i\ d0$	$\equiv$	$\Lambda \| \mathbf{0} \| i\ d0$
	$=$	$\| \mathbf{0} \| [i, d0]$, where $[i, d0] \equiv i'$
	$=$	Snd $[i, d0] = d0$,

Table 2. Invariance of **K** transformation.

$\| \mathbf{K} \| i\ d1\ d0$	$\equiv$	$\Lambda (\Lambda \| \mathbf{1} \|) i\ d1\ d0$
	$=$	$\Lambda \| \mathbf{1} \| [i, d1]\ d0$, where $[i, d1] \equiv i'$
	$=$	$\| \mathbf{1} \| [[i, d1], d0]$, where $[[i, d1], d0] \equiv i''$
	$=$	(Snd ° Fst) i''
	$=$	Snd $i' = d1$,

The principle of interaction gives some *non-symmetry*: there is an object-initiator of action and there is an object-recipient of action. Influence of one object on another is stepwise: it is carried out, if and only if objects are located immediately *beside*. The arrangement happens of two kinds: beside and not beside (distant), and in the second case the objects do not interact. In case of an arrangement beside, the objects immediately enter in interaction. The new object, as a result of interaction, arises and begins its existence -- result of acting, or applying of the first object to the second. Now, if there will be an object located beside thus newly born object, the new act of interaction begins where are two distinct cases.

In the first of them newly generated object captures the existing one, which has appeared beside and acts on it.

In the second case newly generated object is captured by the existing one which affects this object.

In any of these cases the new object arises and begins its existence and this object is considered as a result of such non-symmetrical interaction of two objects-parents. It settles in prestructure on the equal rights with other objects. In particular, this means the following: as soon as the new object-result is generated, it is possible to speak about the new act of interaction.

Thus, the inhabitants of prestructure participate in interaction which evolves by a principle of a dominoe. The following circumstance is important: either there are *initial* atomic objects, or there are *derived* non-atomic objects, each having exactly two ancestors-parents. A question still open: where are the initial objects from, but this discussion will be postponed for a while.

5. Analysis of interaction

The object can be reveled in interaction with other objects if it participates in application. In this case it can show arity, equal to 0 (constant object) or distinct of zero. For simplicity we shall consider a case when the object shows arity, equal to 1 (unary function).

As interaction is carried out through the intermediary -- environment, -- then some metaoperators will be required. For a while, we shall be limited by two metaoperators: Λ -- currying and $\| \bullet \| \bullet$ -- evaluation map.

For any object M we shall check up, whether it can show arity 1 in the environment i. To obtain this we write down

$$\| M \| i\ d0,$$

which represents a value of object M in the environment i. If value of object M shows arity 1 then there is a construction of value of object in the environment

$$\Lambda \| M' \| i\ d0,$$

where M' is the same as object M everywhere, except for a variable to which we should assign the value $d0$: instead of this variable, the number of de Bruijn **0** is written as a prototype of a pointer to $d0$ in environment i'. Environment i' is the same as environment i everywhere, except for an image of this substitutional variable, which is now assigned $d0$:

$$\| M' \| [i, d0].$$

Actually, it was necessary to create a compound metaoperator

$$\Lambda \| \bullet \| \bullet : \text{object} \times \text{environment} \to \text{value},$$

which is object generating, setting up the function of arity 1. Really,

$$\Lambda \| M' \| i\ \ d0 = \| M' \| [i, d0],$$

where $[i, d0] \equiv i'$.

For example, if M is an identity transformation **I** with the characteristic **I** $d0 = d0$ then it is sufficient to assume, that M' is a substitutional variable which is assigned the value $d0$ in environment i (Table 1*)* as was expected.

Here $\Lambda \| \mathbf{0} \| i$ is an image of object **I**, obtained as a result of its interaction with environment. This should be simply a pointer *Snd* to $d0$, located in the modified environment.

Table 3. Invariance of **S** transformation.

|| **S** || *i d2 d1 d0*	≡	Λ (Λ (Λ || **2 0** (**1 0**) ||)) *i d2 d1 d0*
	=	Λ (Λ || **2 0** (**1 0**) ||) [*i*, *d2*]} *d1 d0*, where [*i*, *d2*] ≡ *i′*
	=	Λ || **2 0** (**1 0**) [[*i*, *d2*], *d1*] *d0*, where [[*i*, *d2*], *d1*] ≡ *i″*
	=	|| **2 0** (**1 0**) || [[[*i*, *d2*], *d1*], *d0*], where [[[*i*, *d2*], *d1*], *d0*] ≡ *i‴*
	=	|| **2** || *i‴* (|| **0** || *i‴*) (**1** || *i‴* (|| **0** || *i‴*))
	=	Snd ° Fst ° Fst *i‴* (Snd *i‴*) (Snd ° Fst *i‴* (Snd *i‴*))
	=	Snd ° Fst *i″ d0* (Snd *i″ d0*)
	=	Snd *i′ d0* (*d1 d0*) = *d2 d0* (*d1 d0*),

Other example. If *M* is a cancellator **K** with the characteristic **K** *d1 d0* = *d1* then it is sufficient to assume, that *M′* is a substitutional variable which is assigned the value *d1* in environment *i* (Table 2) as corresponds to the characteristic.

And one more example. If *M* is the allocator **S** with the characteristic

S *d2 d1 d0* = *d2 d0* (*d1 d0*),

then (Table 3) as was expected.

6. Related works

An idea to drop down the computations into self-contained blocks with "nameless dummies" instead of variable was formulated by N. de Bruijn [1, 2] in 1970s. This early formalism appeared rather fruitful to solve distinct and complicated text representation tasks. This approach is known for researchers in theoretical computer science but is not used in modern information technologies (IT). Nevertheless, the related but much earlier ideas of M. Schoenfinkel [3] which were rediscovered by H. Curry [4] are successfully used in many branches of computer science and programming. This approach is known as a *combinatory logic* giving rise to applicative computational systems. But the "natural manner" of computations with *combinators* using then as "wandering" objects was, in fact, not used in modern IT. The natural computations using applicative computational systems were used at the basis of "computational invariants", formulated by V. Wolfengagen [5]. Earlier the idea to use combinators for composing Web-services was used by L. Cardelli et. al [6] but not became widely used in Web science and its applications. The new challenges of using computations with objects were analyzed by H. Barendregt et. al [7], G. Bell et. al [8] and others [9,10] but in relation with applicative computations.

The proposal here is based on the direct use of de Bruijn formalism in connection with combinatory logic to study the main computational effects which are promising in applying the models of natural computations [11, 12]. First, this is a study of object interaction and, second, the study of interaction of object with environment. The generality is in assumption that the environment can be assumed as an object as well. This assumption is distinctive from assumptions in use in area of programming language semantics.

A kind of natural computations was used by B. J. MacLennan [13] in conjunction with the information tasks of a molecular synthesis. At last, the general direction to assume computing as a natural science was argued by P. Denning [14]. Further studying of general ideas of computations especially in connection with applicative computational systems and their spreading was given by L.Yu. Ismailova [15,16].

7. Conclusion and future work

A computational model of natural computations is proposed. It is analyzed from a standpoint of object interaction.

1. A layered structure of computational environment is proposed describing the computational activity of the objects. The simplified assumptions as can be shown lead to a standard semantic model for programming languages. Less standard standpoint is argued to subdivide the environment into "deep" and "peripheral" parts. This gives rise to the natural computations model.

2. Studying the properties of this model shows that it can be represented in a step by step manner which can serve to developing a computational framework for object evaluation.

3. The proposed computational model can be used to discover and verify the various "computational invariants". As was shown the invariants are relative to the computational environment which has a layered structure.

This model can be embedded into host computational model based on applicative pre-structure. The simplicity, flexibility and generality of this model are believed to be useful for semantic and computational analysis of Big Data discovering and using the computational invariants. The basic invariants can be chosen the same as the combinatory basis. The practically sound invariants can be derived using the applicative prestructure at the first stage and applicative structure at the stage of validation.

Acknowledgements:

This research was accomplished at the Institute for Contemporary Education "JurInfoR-MSU" in collaboration with NRNU MEPhI and MIPT under the Project LAMBDA. This work is a generalization of the results, which are associated with the construction of conceptual and computational model obtained at different times during the projects, partially supported by Russian Foundation for Basic Research (RBRF) grants 14-07-00119-a, 12-07-00661-a, 14-07-00072-a, 12 -07-00646-a, 13-07-00716-a, 12-07-00554-a, 14-07-00054-a.

Corresponding Author:
Prof. Viacheslav Wolfengagen
Department of Cybernetics
National Research Nuclear University MEPhI (Moscow Engineering Physics Institute)
Kashirskoye shosse, 31, Moscow, 115409, Russian Federation
E-mail: jir.vew@gmail.com

References

1. de Bruijn NG. Lambda-calculus notations with nameless dummies: a tool for automatic formula manipulation. Indag. Math. 1972;34:381-392.
2. de Bruijn N G. A survey of the project Automath. In: To H.B. Curry: Essays in combinatory logic, lambda calculus and formalism, Academic Press, 1980;579-606.
3. Schoenfinkel MI. Uber die Bausteine der mathematischen Logik, Math. Annalen, 1924; 92:305–316.
4. Curry HB. Functionality in combinatory logic. Proc. National Academy of Sciences of the USA, 1934;20:584–590.
5. Wolfengagen VE. Applicative computing. Its quarks, atoms and molecules, Ed., Dr. Ismailova L. Yu. Moscow: Center JurInfoR, 2010;62.
6. Cardelli L, Davies R. Service combinators for Web computing, HP Labs Technical Reports SRC-RR-148, June 1, 1997;15.
7. Barendregt H, Wiedijk F. The challenge of computer mathematics. Transactions of the Royal Society, 2005;363(1835):2351-2375.
8. Bell G, Dourish P. Yesterday's tomorrows: notes on ubiquitous computing's dominant vision, Personal Ubiquitous Comput., 2007;11(2):33–143.
9. Berners-Lee T, Hall W, Hendler J, O'Hara K, Shadbolt N, Weitzner D. A framework for Web science, Foundations and Trends in Web Science, 2006;1(1): 1-130.
10. Carpenter B. The Internet Engineering Task Force: Overview, Activities, Priorities, ISOC BoT, 2006-02-10, 2006.
11. Wolfengagen VE. Semantic modeling: computational models of the concepts. The Proceedings of the 2010 International Conference on Computational Intelligence and Security, CIS 2010, Sponsors: Xidian University, Beijing Normal University, CPS of IEEE. Nanning, 2010;42-46. doi: 10.1109/CIS.2010.16
12. Wolfengagen VE. Combinatory logic in programming, Ed. Dr. L. Yu. Ismailova. Moscow: Center JurInfoR, 2003;336.
13. MacLennan BJ. Molecular Combinatory Computing for Nanostructure Synthesis and Control. – IEEE Nano 2003, San Francisco, August 12-14, 2003.
14. Denning PJ. Computing is a natural science, Commun. ACM, 2007; 50(7):13-18.
15. Ismailova LYu. Applicative computations and applicative computational technologies. Life Sci J 2014;11(x):-
16. Ismailova LYu. Criteria for Computational Thinking in Information and Computational Technologies. Life Sci J 2014;11(Xs):-

7/5/2014