

73



Applicative Model to Bring-in Conceptual Envelope for Computational Thinking with Information Processes.

Viacheslav Wolfengagen^{*1}, Larisa Ismailova¹, Sergey Kosikov² (jir.vew@gmail.com).

¹NRNU MEPhI, Moscow 115409, Russian Federation, ²NAO "JurInfoR, Moscow 119435, Russian Federation



Viacheslav Wolfengagen

Larisa Ismailova

Sergey Kosikov

Your
Logo
Here

ABSTRACT

The present moment, in fact, is a transition from the concept of computer science as an artificial science to the understanding that information processes are abundant in nature. All object constructions are considered as virtual objects, which are possibly combinators. They are connected by "combinatory glue" and form an applicative prestructure. The defended arguments boil down to the following: (1) the computation cannot be eliminated not only from the research method, but also from the subject of the study - what is being studied and (2) the real value of computer science lies in those conclusions that experts can draw based on their expertise using a fairly rich and deep system of reasoning. For example, the most fruitful Karoubi envelope features for computing are discussed.

INTRODUCTION

In modern research, computing is understood as the study of natural and artificial information processes, which, in turn, are recognized as ubiquitous.

Information processes are manifested through a variety of external forms, based on computational communication. Communication consists in the fact that the parties involved exchange process messages, and the same process message can be expressed in different sentences and vice versa, the same sentence can express different process messages. The ability to identify the same process in different sentences is based on the hypothesis of the stratification of each communication language into two languages: -- a genotypic language for unambiguous recording of the content of processes; -- a phenotypic language with a variety of means for expressing the same process. The implementation of the connection of both languages by embedding in an applicative computational model equipped with a convertibility relation is proposed.

METHODS

One of the goals of this work is to revise the well-known ideas that how to get the value of an expression using an applicative computing system.

Computations and/or symbolic transformations have undoubtedly become one of the dominant trends in modern computer research. In particular, this refers to the implementation of the processes of the derivation of an object with given properties, which is studied with the formation of sets of equalities, which together lead to a computational model. At the same time, the comprehensive theory of computations receives a targeted equational description, making the process of studying its properties and capabilities rather constructive.

ENTER FINDING #1

Table 1. Computational envelope for applicative Eval-system

Equation	Notation	Principle
$B = \Psi(\Phi I)$	$(\cdot) =$	Evaluation of an application
$CB'D = \Psi(\Phi D)$	$([\cdot, \cdot] =)$	Evaluation of an ordered pair
$VK\rho = K = KK\rho$	'encapsulation	Evaluation of a constant
$Curry = \Psi(\Phi D)$	$([\cdot, \cdot] =)$	Evaluation of an ordered pair

ENTER FINDING #2

4.1 Software glue

So we got to the semantic side, which is already obvious to any programming person. Indeed, in the semantics of programs, as the main premise, it is believed that all the information about the state of computation is contained in the environment. The program, however, seems to be composed, generally speaking, of variables between which relations, or links are established, which, in turn, can be variables. If relations are not variables, then they act as constant links -- they are program glue, organizing the interaction of variables.

How does a variable exhibit itself? Only if it is transferred. . . to the excited state. According to current notions of programming, this refers to assigning a variable value [that is, substitution?].

Thus, we come to the conclusion that transitions between states are generated by assignments: assignment changes the "computational potential" of objects, moreover, jump stepwise, and the assignment result is "captured" by the environment of [computations], which thus passes in new state. The semantics of the computations is reduced to the description of all possible admissible transitions between states. Conceivable states include both allowed and forbidden states. In normal mode the alternation of states caused by assignments, apparently, leads only to allowed states, but in a number of special cases of assignment undesirable situations arise that we have not learned to use to our advantage. They are usually classified as prohibited states, and they are processed in special ways. Let's say an example would be overflow of a bit grid, etc.

Some preliminary result: the "assembly" of objects is variables and software glue, and the computation process exhibits itself precisely through variables. At least we do not yet know another way. The most widely used way of influencing variables is to assign values to them.

ENTER FINDING #3

..., Molecules, Atoms, ... of Computations

How to make the object begin to exhibit its state? The answer to this question is known, and it consists in the transition from objects to their values.

Does the object exhibit itself in an isolated form, and what is meant by its computational

behavior? If the object is atomic, then the answer is no. If the object is molecular, then in this case the situation is as follows:

- one of the component objects is applied, or, in other terminology, affects another, and the result is considered the result of such an impact. Value in the sense that the acting object plays the role of a function, and the affected object acts as an argument [of this function]. The result of the action is considered, expressed in mathematical language, the value of the function on the argument presented to it; - but this is aggregated value, and it does not say anything about the state, or, so to speak, computational potential of neither a function object nor an argument object;

- if we talk about differentiated value, then a certain context will be required. Canonically in the theory of programming as a context chosen is the computation environment.

DISCUSSION

By cognitive interference we mean the loss of the information process that occurs when ideas/processes generated by other participants interfere with the activity of generating their own ideas belonging to this individual [8], [15]. In all cases, certain models and capabilities of cognitive modeling [22], [19] play a decisive role. Cognitive interference occupies a territory on the border between personality and intelligence. Intelligence is derived from how people perform certain types of tasks. Poor performance, however, does not necessarily mean low intellectual potential; it may be because the person was upset, thinking about something good or unmotivated. All of these circumstances can contribute to cognitive interference: thoughts that invade task-related activities and serve to reduce the quality and level of productivity. Some cognitive interferences can be considered as aspects or products of the individual, as they are associated with personal concerns that interfere with attention to the task. A personality can help increase productivity (for example, due to high motivation and the ability to immerse yourself in tasks), but it can also weaken it (for example, due to concern about the consequences of failure and refusal to work with a tester).

CONCLUSIONS

1. Varying from research to research, the notion of the essence of an 'object' can be expressed from a computational point of view, presenting and embedding it in some kind of primitive frame. The scheme operates within the framework of function theory related to combinatory logic and generates the original conceptual envelope.
2. Combinators provide a semantically consistent basis to build a model of data objects. Used objects inherit both syntactic and semantic features and features of the original representation of the object. This leads to computations of the form object-as-functor and generates an improvement process that allows one to take into account methods for specific objects.
3. As can be shown, concepts are embedded in the envelope and inherit the logical properties of objects. The theory of higher orders is used (with definite descriptions).
4. The improvement process satisfies equationally expressed conditions. Separate methods should be studied in an equational framework.

ACKNOWLEDGMENTS & REFERENCES

This research is supported in part by the Russian Foundation for Basic Research, RFBR grants 20-07-00149-a, 19-07-00326-a, 19-07-00420-a, 18-07-01082-a.