Gnoseology-based Approach to Foundations of Informatics

Mykola Nikitchenko

National Taras Shevchenko University of Kyiv
01601, Kyiv, Volodymyrska st, 60
nikitchenko@unicyb.kiev.ua

Abstract. Now it is generally recognized that informatics is to be based on its own solid foundations which should state its self-dependence and provide its self-development. In the paper we propose to use a gnoseology-based approach for developing methodological, conceptual, and formal levels of foundations. At the methodological level we describe a number of general gnoseological principles and a system of philosophical categories specifying the main features of the approach. On the conceptual level we propose to elucidate basic notions of informatics in integrity of their intensional and extensional aspects. Then we use developed notion models to construct their mathematical counterparts at the formal level. Such constructions start with formalization of the notion of data as intensionalized data. The main kinds of such data are presets, sets, and nominats (nominative data). Then we define the notion of abstract computability applicable to the described types of intensionalized data. At last, we construct hierarchy of predicate logics over introduced intensionalized data.

Keywords: foundations of informatics, methodology, intension, extension, intensionalized data, abstract computability, predicate logics.

1 Introduction

Informatics is a relatively young discipline. As such, it borrows its foundations from disciplines it is based on, primarily from mathematics, linguistics, logic, and philosophy. But coming into mature age, informatics is searching for its own foundations, which should state its self-dependence and provide its self-development.

Constructing of foundations is a highly challenging problem. This is caused by the diversity of topics studied in informatics, by variety of approaches to such topics, by vagueness of notions specified for such topics, etc.

In the first approximation we treat foundations of informatics as the systematic study of basic notions (concepts) of informatics taking in the integrity of their essential aspects; as scientific inquiry into the nature of information-related theories, their scopes, properties, and limitations.

Speaking about foundations we cannot avoid discussions about methodological (philosophical) principles of developing such foundations. It means that a certain philosophical position should help to specify methodological principles for

constructing foundations of informatics. We will try to demonstrate that many principles of gnoseology confirm their usefulness for elucidating of basic notions of informatics. Thus, we will advocate that a gnoseology-based approach can help in constructing foundations of informatics.

The aim of this paper is to describe the main principles of gnoseology-based approach and show how it can be used to explicate the basic notion of informatics – the notion of data – in integrity of its intensional and extensional aspects; and to develop fragments of mathematical formalisms, namely, computability theory and mathematical logic, oriented on such conception of data. This aim determines the structure of the paper: first, we describe methodological principles of our approach for developing foundations of informatics; then we specify the notion of intensionalized data and define basic computability theory over such data; and, at last, we describe logics of partial predicates oriented on such data.

2 Methodological Principles of the Approach

We propose to identify three levels of foundations specified by the following principle.

Principle of three-level foundations of informatics: foundations of informatics should be constructed in integrity of methodological (philosophical), conceptual (scientific), and formal (mathematical) levels.

What methodological theory should be used at the first level? There can be several ways to answer this question. The first is ad hoc one which specifies methodological principles only when it is required by some special topics. The second is specialized one. It introduces only methodological principles specific for informatics and uses them as a basis for theory development. And the third tries to integrate general (philosophical) methodological principles with special ones and present them in an explicit form.

Our approach [1] is the third one. We base it on a philosophical system developed by Hegel [2]. Strictly speaking, we use only a gnoseological component of his system. Gnoseology (epistemology) is a theory of cognition, which aims to examine the nature of knowledge, its limits, how it is derived, and how it is to be validated and tested. Thus, our approach is *a gnoseology-based approach*. This is to be supported by *the principle of gnoseology-based theory*: a theory should be developed according to the main principles, laws, and methods of gnoseology. This principle is a weak form of a Hegel's idea of a theory as applied logic. The central part of gnoseology is a system of philosophical categories. These categories can be considered as the most general features, characteristic of things. Examples are: *subject* and *object*; *abstract* and *concrete*; *internal* and *external*; *quality*, *quantity*, and *measure*; *essence* and *phenomenon*; *individual*, *general*, and *particular*; *whole* and *part*; *content* and *form*; *cause* and *effect*; *goal* and *tool*; etc. [2].

Thus, the principle of three-level foundations identifies three types of notions – categories, scientific notions (concepts), and formal notions – that constitute the basis of each level respectively.

Other general methodological principles can be briefly explained by the following considerations. Any object of study has numerous connections (relations, aspects) with other objects, thus, it can be considered as the totality of all its aspects. But we cannot immediately investigate this totality, therefore we start with some aspects (chosen due to abstraction); then we study some other aspects with their relations to the first aspects (thus making concretization). These aspects are considered as essential ones and are chosen according to practical and theoretical criteria. Despite the simplicity and coarseness of the above considerations they lead to the following principles of gnoseology.

Principle of development from abstract to concrete: development is definitely oriented change of the object (notion) from abstract to concrete (from simple to complex, from a lower level to a higher one, from the old to the new).

Triadic principle of development: one of the main schemes of development is specified as a development triad: *thesis – antithesis – synthesis*.

Principle of unity of theory and practice: theory and practice should be considered as influencing each other. This principle substantiates development of informatics notions in praxeological perspective, i.e. this development should be based on analysis of human action in information domain. The praxeological aspect is one of the main philosophical aspects relating categories of subject and object. It lies in one line with ontological, gnoseological, and axiological aspects.

Let us note that the importance of philosophical foundations for information-related disciplines (such as information science) is widely recognized. Different philosophical systems were proposed to use for this purpose, for example, K. Popper's ontology in [3], philosophy of Kuhn and Peirce in [4]. There are also proposals to develop a specific epistemology for information science [5]. A short description of philosophical approaches can be found in [6, 7]. We also advocate the necessity of philosophical studies oriented on informatics. Actually it means that an interdisciplinary approach should be used for developing foundations of informatics.

At the second, conceptual level of foundations, methodological principles are used to develop the basic scientific notions and their interrelations. Such system of basic notions (ontology) presents structure and properties of a domain under investigation. Developing scientific notions (concepts) of informatics, we chose the "closest" categories and "project" them on such concepts. Such projections transfer properties of categories and their relationships from philosophical onto conceptual level.

Mathematical base of informatics is formed by set theory, universal algebra, mathematical logic, and computability theory. Many important and useful results for informatics were obtained on this base. Still, some discrepancies between above mentioned disciplines and problems of informatics can be also admitted. They concern questions of defining data structures on various levels of abstraction, computability of functions over such data, processing of data with incomplete or fuzzy information, construction of logics oriented on information processing, etc. Such discrepancies require additional efforts in modeling problems of information domain with existing mathematical formalisms. Therefore it is reasonable to formulate a problem of developing own, more adequate mathematical foundations for informatics. But what should be the starting point of such development?

Analysis of existing mathematical formalisms shows that they are constructed on a set-theoretic platform. But its main notion – the notion of set – is explicated in

extensional style. This style is supported by the very first axiom of set theory – the extensionality axiom: two sets are equal if they consist of the same elements [8]. N. Bourbaki in his numerous treatises aimed to write a thorough unified account of all mathematics based on extensional set theory. At that period the extensional approach played a positive role permitting to specify formally many properties of mathematical objects. But we can see now more and more facts when a pure extensional orientation becomes restrictive for further development of informatics.

Thus, we propose to add explicit intensional component to notions and construct them in integrity of intensional and extensional aspects. Here the intension of a notion (of a concept) means properties which specify that notion, and the extension means objects which fall under the notion, i.e. have the properties specified by the notion intension. Intension and extension of a notion we consider as projections of categories *general* and *individual* respectively.

We should admit that this proposal is not new. The distinction between intensional and extensional aspects of a notion was known from ancient times. Aristotle in his *Posterior Analytics* already specified this distinction though he did not use explicitly the above terms. Many logicians since that time examined the questions of intension/extension dichotomy. A second wind to these investigations was given by G. Frege with his famous *meaning triangle* and R. Carnap with his *intensional/extensional investigations*. Though the dichotomy under discussion was studied primarily in logic, semiotics, and linguistics, last years it was also investigated in informatics. In its branches related to artificial intelligence, data and knowledge bases, semantic web, etc., the intensional aspects now play an important role. But in formalized (mathematical) theories intensional aspects are still used very restrictively.

The above presented considerations advocate the following principle: a notion should be presented by the triad *notion intension – notion extension – integrity of intension and extension; the intension in this integrity play a leading role* (*the principle of triadic model of a notion*). This principle may be considered as an enhancement of Frege's meaning triangle.

At the formal level of foundations, the notions, constructed at the previous level, are specialized in order to get their reasonable formalization. This formalization should take into account intensional and extensional notion aspects. This level is important for informatics because formal notions provide a basis for automatization of various phases of information processing.

Having specified a three-level structure of foundations and main principles of theory development, we can now consider basic notions of informatics at the conceptual level. As the name shows, the main notion of informatics is the notion of *information*. It can be formalized in various aspects [9, 10], but the most important ones are aspects represented by the philosophical categories of *form* and *content*. This understanding is supported by the etymology of 'information', derived from Latin '*informare*': "to give form to". Forms of information which are relatively independent of its content we call *data*.

Thus, we get the initial part of the well-known "data – information – knowledge – wisdom" hierarchy (DIKW-hierarchy) [11]. These concepts are the building blocks of informatics; and explications should be developed for all these notions. This is a difficult challenge, therefore it is reasonable to start with the notion of data which lies in the base of DIKW-like hierarchies, is relatively simple, and is an appropriate

subject for further formalization. The word 'data' is used in this paper as a plural and as a singular noun.

Among many characteristics of data (see [9] for a detailed discussion) we choose "data as manipulable objects" [12]. This treatment is a projection on conceptual level of the category of *object* taken in the praxeological aspect.

So, our abstract understanding of the notion of data includes three moments only: 1) data are regarded as a form of information content; 2) data are (relatively) independent from information content; 3) data are manipulable objects. These moments demonstrate that we treat data in a very broad sense. The principle of development from abstract to concrete suggests that other data characteristics should be introduced on the later stages of development.

According to the principle of triadic model, we treat data as integrity of their intension and extension, thus we enrich traditional understanding of data with intensional component [13]. Obtained data are called *intensionalized data*. Let us admit that there is an analogy with the notion of *typed data*, but the latter is usually understood in the extensional sense while we aim to emphasize intensional features of our approach.

3 Intensionalized Data

To simplify formalization of the notion of intensionalized data, we start with the most abstract understanding of data as some objects (keeping in memory that on later stages data should be considered as manipulable objects representing information content). The first step in objects explication is classification of intensions which can be prescribed to objects. Such classification can be made with respect to various criteria. Our gnoseology-based approach suggests to develop a classification induced by some categories. As such we choose categories *whole* and *part* which are among the first categories revealing the category of *essence* [2]. Thus, we introduce two different intensions I_W and I_P: object as a whole (unstructured object) and object with parts (structured object) respectively.

Further development of I_W is done in accordance with categories of *abstract* and *concrete*. One of projections (restrictions) of these categories on the conceptual level describes abstract as less informative than concrete. In extreme cases, an object can be regarded as a "black box" (intuitively it means that nothing is "visible", and therefore nothing is known about object) or as a "white box" (everything is "visible" and recognizable). Thus, we articulated new intensions I_{WB} and I_{WW} respectively. An intermediate intension is denoted by I_{WBW} ("black or white box"). The introduced intensions describe the main possibilities to treat object as a whole.

To come to richer intensions we should treat objects as structured (with intension I_P). In this case we get a triad: *whole – part – structure*, where structure is the synthesis of categories *whole* and *part*. Now we will invent properties (intensions) of object structures. The development principle stimulates us to start with simple structures. Simplicity means that all parts of an object are recognized and fixed. In this case each part can be regarded as a whole. Relations within the object are also recognized and fixed. The next question is: what intensions can be prescribed to the

parts and relations? Being the wholes, parts can have intensions of black and/or white boxes. Should it be allowed for relations to have the same three intensions? It is not reasonable to do this at the first stages of data development; therefore we prescribe to relations intensions of white boxes only. The above specification of object structure permits to call it *hard structure*. Thus, we divide the intension I_P into two sub-intensions I_{PH} and I_{PS} specifying objects with hard and soft structures respectively. In this paper we restrict ourselves by studying objects with prescribed intension I_{PH} only which is simpler than I_{PS}.

We continue with I_{PH} classification that is caused by possible relations between object parts. Such relations are classified along the line *tight–loose*. Loose relations mean that parts are not connected with each other (in Hegel's words, are indifferent to each other); tight relations mean that parts are connected. Thus, new intensions I_{PHL} and I_{PHT} are articulated.

Parts of objects with intension I_{PHL} and I_{PHT} are usually called *elements* and *components* respectively. Considering elements as wholes, we can treat them with intensions of black and/or white boxes. Three new intensions which are sub-intensions of I_{PHL} stem from this: I_{PHLB}, I_{PHLW}, and I_{PHLBW}.

It is worth to discuss objects (data) with these intensions in more detail. Objects with intension I_{PHLB} should be regarded as collections of black boxes because they consist of clearly separated elements with unknown interior (content) that have no relations with each other. Such objects we call *presets*. For example, buying several tickets of an instant lottery someone gets a preset because surfaces of the tickets are covered by opaque material making them black boxes. Collections of elements which are white boxes (intension I_{PHLW}) are called *explicit sets*. Collections with intension I_{PHLBW} contain "black" and "white" elements (*mixed presets*). For example, when someone comes at a party he first classifies people as known to him (white elements) or unknown (black elements). Here we should note that richer intensions than I_{PHLB} and I_{PHLBW} can allow extracting additional information from "black" elements (and in this case they become "white" or "gray" elements).

Thus, we have articulate three kinds of objects: presets, sets, and mixed presets. Now just sets serve as a basis for formalization of informatics notions, but it seems reasonable to use also the notions of preset and mixed preset for this purpose. Their importance can be substantiated by the necessity of defining data at various abstraction levels that cannot be adequately captured by the extensional notion of set. Therefore in is not strange that numerous attempts were made for constructing set theory without extensionality (see, for example, [14]). We stop further classifications of objects with intension I_{PHL} (I_{PHL}-objects) and start classifying I_{PHT}-objects.

Components of I_{PHT}-objects are in some way related to each other, so, contrary to the elements of I_{PHL}-objects, the components are not allowed to permute freely with each other within I_{PHT}-objects. Examples are: lists, trees, graphs, arrays, etc. It seems (and it is true) that a lot of useful kinds of I_{PHT}-objects can be considered. This fact poses a question what sub-intension of I_{PHT} should be introduced first? And again we appeal to the principle of development which advises to start with the simplest relation between components. Such a relation connects only two components. In this case these components look as opponents, and the relation should link them, thus making their synthesis. From this follows that it is reasonable to treat the first component as a black box, the second as a white box, and their synthesis as a "dipole"

consisting of the black and white boxes. To make these abstract considerations more concrete we should involve practical observations of activities in the information domain which prompts us that the white box is a *name* of the black box; and their relation is a *naming* (*nominative*) relation. Thus, described objects are presets whose elements are named values. We propose to call such objects *nominats* and denote corresponding intension as I_{ND}. In Slavic languages the term 'nominat' has two different meanings: a naming expression or a value of such expression. Our proposal unites these meanings, because nominat is a unity of names and values. Nominats are also called *flat nominative data*.

Traditionally, notations of functional style are chosen to represent nominats. For example, a nominat with names $v_1, \ldots, v_n$ and values $a_1, \ldots, a_n$ respectively, is denoted by $[v_1 \mapsto a_1, \ldots, v_n \mapsto a_n]$. If values themselves are nominats, then we get the notion of hierarchic nominats (hierarchic nominative data); for example $[v_1 \mapsto [u_1 \mapsto b_1, \ldots, u_k \mapsto b_k], \ldots, v_n \mapsto [t_1 \mapsto c_1, \ldots, t_m \mapsto c_m]]$ is a 2-level nominat.

It is important to admit that nominats can model the majority of data structures used in informatics [1]. For example, a set $\{e_1, \ldots, e_m\}$ can be represented as $[1 \mapsto e_1, \ldots, 1 \mapsto e_m]$, where 1 is a standard name which have different values $e_1, \ldots, e_m$; a tuple $(e_1, \ldots, e_m)$ can be represented as $[1 \mapsto e_1, \ldots, m \mapsto e_m]$ with 1,…, m as standard names; a sequence $<e_1, \ldots, e_m>$ can be represented as $[1 \mapsto e_1, 2 \mapsto [\ \ldots, 2 \mapsto [1 \mapsto e_m, 2 \mapsto \varnothing_n]\ldots]]$, where 1, 2 are standard names and $\varnothing_n$ is the empty nominat.

Summing up, we can conclude that the developed notion of intensionalized data can represent data structures used in informatics, and besides, this representation looks richer and more adequate than traditional set-theoretic representation.

In this section we have concentrated only on classification of object intensions ignoring operations with objects. To come back to the initial treatment of data as manipulable objects, we should describe operations allowed for data with different intensions. Actually it means that we should try to construct basic computability theory for intensionalized data. This will be done in the next section.

4 Computability over Intensionalized Data

We begin with general considerations about traditional computability. Such computability is usually understood as computability of *n*-ary functions defined on integers or strings. It may be called Turing computability. In the light of our investigations traditional computability does not pay much attention to variety of data intensions, because it concentrates on computability over integers (or strings) which have fixed intensions. But in informatics other data structures with different intensions are also used; therefore for these structures a new notion of computability is required [15].

The computability problem is not the only aim of our investigations. Now it is generally recognized that information systems should be developed successively from abstract specifications via more concrete representations up to detailed implementations in chosen programming languages. And it is important to connect computability with stages of system development. We intend to introduce such a

unified notion of computability that can be applied to every stage of system development and can be easily transformed when moving from stage to stage. Such a kind of computability should be applicable to data structures of different abstraction levels and is called *abstract computability*. A partial case of this computability, oriented on intensionalized data, is called *intensionalized computability*. The idea behind it is the following: for data processing it is allowed to use only those operations that conform to their intensions. Thus, intensionalized computability is *intensionally restricted computability*. In fact, such computability is a *relative computability* – relative to data intensions. Usually it is required that data have *finite structures*; a corresponding data intension we denote by I_{PHF}.

There is a difficulty in defining intensionalized computability that is caused by the fact that for finite structured data with intension I_{PHF} we do not have precise definitions of their components and relations between components; thus, precise definition of computability is not possible. Of cause, we can introduce data with precisely described sub-intensions of I_{PHF} (like intensions for tuples, list, trees, etc.) and then define computability for such specific intensionalized data. But in this case we will not get a unified computability theory for intensionalized data. To overcome this difficulty we propose to apply the method of reduction of intensionalized computability to traditional Turing computability. To do this we will first define a special form of finite structured data with a fixed intension, and then reduce data with other intensions to this special form.

Let D be a class of data with intension I_D. Such class is also denoted as $[I_D, D]$, data from this class are called I_D-intensionalized data or simply I_D-data. Assume that we treat data from D as finite structured data. Our intuitive understanding of a such data is the following: any such data d consists of several basic (atomic) components $b_1, ..., b_m$, organised (connected) in a certain way. If there are enumerably many different forms of organisation, each of these data can be represented in the (possibly non-unique) form $(k,<b_1, ..., b_m>)$, where k is the *data code* and the sequence $<b_1, ..., b_m>$ is the *data base*. Data of this form are called *natural data* [1]. More precisely, if B is any class and Nat is the set of natural numbers, then the class of *natural data* over B is the class $Nat(B) = Nat \times B^*$. We use the term 'class' for collections of intensionalized data; term 'set' is used for collections which intensions are sub-intensions of sets. As finite structured data can have different representations, we should introduce multi-valued functions for constructing such representations. Function f is *multi-valued* (non-deterministic) if being applied to the same input data d it can yield different results during different applications to d (i.e., function's graph is not a functional relation). To avoid complex notations with subscripts, to will denoted a class of partial multi-valued functions over D as $D \to\to D$. A multi-valued function is *injective*, if it yields different values on different arguments.

Now we are ready to give the formal definition of a class of intensionalized data with some intension I_D which is a sub-intension of I_{PHF}. A class D is called a class *of finite structured data*, if a class B and a total multi-valued injective mapping *nat*: $D \to\to Nat(B)$ are given. This mapping *nat* is called the *naturalization* mapping. Naturalization mapping is actually an *analysing* mapping: it finds in a data d its components and their interrelations according to the properties of data prescribed by its intension. Dually to *nat* we introduce *denaturalization* mapping *denat* which reconstructs (*synthesizes*) data of class D from natural data. For simplicity's sake we

assume that $denat=nat^{-1}$. Denaturalization mapping is a partial single-valued mapping. Naturalization and denaturalization mapping are also called *concretization* and *abstraction* mappings respectively.

Introduction of naturalization mapping is a crucial moment for defining intensionalized computability. This mapping can be regarded as a formalization of data intension; and this enables us to reduce an intuitive notion of intensionalized computability over D with intension I_D to formally defined *natural computability* over D. The latter is then reduce to a new special computability over $Nat(B)$ that is called *code computability*. To define this type of computability we should recall that in a natural data the code collects all known information about data components. Thus, code computability should be independent of any specific manipulation (processing) operations of the elements of B and can use only information that is explicitly exposed in the natural data. The only explicit information is the data code and the length of the data base. Therefore in code computability the data code plays a major role, while the elements of the data base are treated as black boxes which virtually do not affect the computations. These elements may be only used to form the base of the resulting data. To describe the code of the resulting data and the order in which elements of the initial base are put into the base of resulting data, a special function of type $Nat(B)\to\to Nat(B)$ should be defined. Such a function is called *index-computable*. These considerations lead to the following definition.

A function g: $Nat(B)\to\to Nat(B)$ is called *code-computable* if there exists an index-computable multi-valued function h: $Nat^2\to\to Nat \times Nat^*$ such that for any k, $m\in Nat$, $b_1,..., b_m\in B$, $m\geq 0$, we have $g(k,<b_1,...,b_m>) = (k',<b_{i_1}, ..., b_{i_l}>)$ if and only if $h(k,m) = (k',<i_1,...,i_l>)$, $1\leq i_1\leq m,..., 1\leq i_l \leq m$, $l\geq 0$. If one of the indexes $i_1,...,i_l$ lies outside the interval $[1, m]$, or $h(k,m)$ is undefined, then $g(k,<b_1,...,b_m>)$ is also undefined.

In other words, in order to compute g on $(k,<b_1,...,b_m>)$, we have to compute h on (k,m), generate a certain value $(k',<i_1,...,i_l>)$, and then try to form the value of the function g by selecting the components of the sequence $<b_1,...,b_m>$ pointed to by the indexes $i_1,...,i_l$.

It is clear, that index computability of h: $Nat^2\to\to Nat \times Nat^*$ may be reduced by traditional methods of recursion theory to computability of a certain partial recursive function r:$Nat\to\to Nat$.

We are ready now to give the main definition of this section. A function f: $D\to\to D$ is called *naturally computable* (with respect to given B and *nat*) if there is a code-computable function g: $Nat(B)\to\to Nat(B)$ such that $f = denat \circ g \circ nat$.

The class of all naturally computable functions is denoted by $NatComp(D, B, nat)$.

Thus, intensionalized computability is defined via a sequence of the following reductions: intensionalized computability – natural computability – code computability – index computability – partial recursive computability. Analysing the definitions, we can also conclude, that natural computability as a generalization (relativization) of enumeration computability. In fact, for $B = \varnothing$ code computability is reduced to partial recursive computability on Nat, and natural computability is reduced to enumeration computability (with respect to *nat*). Therefore, the notions of code and natural computability defined above are quite rich.

Having defined the notion of natural computability, we can now construct algebraic representations of complete classes of naturally computable partial multi-valued functions for various kinds of intensionalized data. In this short paper we give without details only few examples. We start with the simplest case.

Let D be a preset with prescribed intension I_{PS} ($=\mathtt{I}_{\mathtt{PHLB}}$). It means that nothing is known about its elements. This treatment can be formalized by the naturalization mapping $\mathrm{nat}[\mathrm{I}_{PS}]$: $D \to\to Nat(D)$ such that $nat[\mathrm{I}_{PS}](d)=(0,<d>)$ for every $d \in D$. To define the complete class of naturally computable functions over $[\mathrm{I}_{PS}, D]$, we have to describe all index-computable function of the type h: $Nat^2 \to\to Nat \times Nat^*$. It is easy to understand that under the naturalization mapping $nat[\mathrm{I}_{PS}]$ we need to know the results of index-computable function only on the element (0,1). On this input data an index-computable function 1) can be undefined, 2) can yield (0, 1), or 3) can make non-deterministic choice between being undefined and yielding (0, 1).

These three cases induce the following functions of type $D \to\to D$: 1) the everywhere undefined function *und*, 2) the identity function *id*, and 3) the non-deterministic function *und-id* such that *und-id*(d) is undefined or is equal to d.

It means that the following result was proved: *the complete class of naturally computable partial multi-valued functions over* I_{PS}*-intensionalized preset D consists of functions und, id, and und-id.*

In other words, the three functions defined above are the only computable function over "black box" intensionalized data.

The next example describes computability over subclass of hierarchic nominats. This subclass $NAD(V,W)$ is called the class of *named data* and is defined inductively on the basis of a finite set of names $V=\{v_1, \ldots, v_{\mathrm{m}}\}$ and a preset of basic values W:

1) If $w \in W$, then $w \in NAD(V,W)$,
2) If $v_1,\ldots,v_{\mathrm{n}}$ are pairwise distinct names from V, $d_1, \ldots, d_{\mathrm{n}}$ are from $NAD(V,W)$, then $[v_1 \mapsto d_1, \ldots, v_{\mathrm{n}} \mapsto d_n]$ belongs to $NAD(V,W)$.

The intension of such data is denoted by I_{NAD}. This understanding of $NAD(V,W)$ can be represented by the naturalization mapping $nat[\mathrm{I}_{NAD}]$: $NAD(V,W) \to\to Nat(W)$ which is defined inductively as follows:

1) if $d \in W$, then $\mathrm{nat}[\mathrm{I}_{NAD}](\mathrm{d})=(\mathrm{c}(0,0),<\mathrm{d}>)$;
2) if $d = [v_{i_1} \mapsto d_1,\ldots, v_{i_n} \mapsto d_n]$, $i_1<\ldots<i_n$, $n \geq 0$,

$nat[\mathrm{I}_{NAD}](d_j)=(k_j,<b_{j1}, \ldots, b_{jl_j}>)$, $1 \leq j \leq n$, then

$nat[\mathrm{I}_{NAD}](d) = (c(1, c(n,c(k'_1,\ldots c(k'_n,0)\ldots))),<b_{11},\ldots,b_{1l_1}, \ldots,b_{n1}, \ldots, b_{nl_n}>)$,

where c: $Nat \times Nat \to Nat$ is the Cantor's pairing function; $k'_j = c(i_j,c(k_j, l_j))$, $1 \leq j \leq n$.

Having defined the naturalization mapping for $[\mathrm{I}_{NAD}, NAD(V,W)]$, we obtain the class $NatComp(D, B, nat[\mathrm{I}_{NAD}])$ of all naturally computable functions over the class $NAD(V,W)$. As the basic functions from this class we choose operations [15] of *naming* $\Rightarrow v$, *denaming* $v \Rightarrow$, and *checking* $v!$ with name $v \in V$ as a parameter; we also use non-deterministic *choice* $\Diamond$ which on d yields d or $\varnothing_{\mathrm{n}}$. The main operations over this class of functions (we call them *compositions*) are *multiplication* $\circ$ (functional composition), *conditional iteration* * (while–do), and *overriding* ∇.

The following theorem which gives an algebraic description of the class $NatComp(D, B, nat[\mathrm{I}_{NAD}])$ can be proved: *the complete class of naturally computable*

partial multi-valued functions over the I_{NAD}*-intensionalized class of named data NAD(V,W) coincides with the class of functions obtained by closure of functions* $\Rightarrow v$, $v\Rightarrow$, $v!$, *and* $\Diamond$ *under compositions* $\circ, *$, *and* $\nabla (v \in V)$.

The last example will describe computability over the I_{SEQ}-intensionalized class *Seq*(*B*) of sequences constructed hierarchically over a preset *B*. The structure *Seq*(*B*) has been investigated in different works. *W*e shall use the notations of [16]. The following functions are introduced: *first*, *tail*, *apndl*, and *is-atom*. Also, we need a composition, called *construction*: $[f,g](d) = \langle f(d), g(d)\rangle$ (*d* belongs to *Seq*(*B*), *f* and *g* are functions over *Seq*(*B*)). The following theorem can be proved: *the complete class of naturally computable partial multi-valued functions over the* I_{SEQ}*-intensionalized class Seq(B) coincides with the class of functions obtained by closure of functions first, tail, apndl, is-atom, and* $\Diamond$ *under compositions* $\circ, *$, *and* $[]$.

Having introduced in this section the notion of intensionalized computability, we actually defined those operations which are allowed to apply to such data. Thus, combining definitions of intensionalized data with definitions of computable functions over such data, we made explication of data as manipulable objects. To reason about intensionalized data, we should develop special logics oriented on such data. This will be done in the next section.

5 Predicate Logics over Intensionalized Data

The main idea of developing logics over intensionalized data consists in defining such logical constructs (connectives, quantifiers, etc.) that conform to the data intensions. It means that these logical constructs (to be semantically explicated as compositions of predicates) should use only such data information that is specified by data intensions.

To make these intuitive considerations more strict, we start with constructing a semantic base for intensionalized logics.

Let $[I_D, D]$ be a class of intensionalized data. A class of partial functions $P = D \rightarrow Bool$ is called a class of *partial predicates* over *D* (*Bool* = {*T*, *F*}). Operations over *P* are called *predicate compositions*. Let us admit that we do not restrict predicates by data intensions. This is necessary in order to have a wider class of models for logics. But data intensions should restrict the class of predicate compositions; from this stems the main problem of logic development: how to define compositions which are intensionally restricted by I_D?

We will define such compositions according to the principle of development from abstract to concrete. Therefore we should start with the most abstract intension I_B ("black box" data). The basic compositions defined in this case are compositions of predicate disjunction $\vee$ and negation $\neg$. We define these compositions in the style of strong Kleene's connectives.

Let *p* and *q* be predicates, *d* be from *D*. Then $(p \vee q)(d)$ is defined and equal to *T* if $p(d)$ or $q(d)$ is defined and equal to *T*; is defined and equal to *F*, if both $p(d)$ or $q(d)$ are defined and equal to *F*; and is undefined in all other cases. The value $\neg p(d)$ is a dual to the value of $p(d)$. Other propositional compositions can be defined analogously.

From these definitions we see that logics over $[I_B, D]$ are just different variants of propositional logics (partiality should be also taken into account). A semantics base of such logics are predicate algebras of the form $<D \to Bool, \vee, \neg>$. Properties of such algebras will specify calculi for our logics.

We will not describe logics over classes of data with intensions I_W and I_{BW}, because the intension I_W specifies only one concrete class of data, and its logic is only a logic of this class (singular logic); logics over I_{BW}-intensionalized data are combinations of propositional and singular logics.

The next data intension we consider here is the intension of (infinite) flat nominats I_{ND} which is a sub-intension of I_B. Data, intensionalized with I_{ND}, have a form $[v_1 \mapsto a_1, v_2 \mapsto a_2, \ldots]$. In traditional logic names $v_1, v_2, \ldots$ are called individual variables, and data from D are called variable assignments, variable valuations, etc. In this case $D = {}^{V}A$, where V is a set of individual variables, A is a class of individual values, and ${}^{V}A$ can be considered as a class of partial function from V to A. Predicates from ${}^{V}A \to Bool$ are called *quasi-ary predicates.* At the level of I_{ND}-data, additionally to propositional compositions, we can define a new composition of *renomination* $R^{v_1,\ldots,v_n}_{x_1,\ldots,x_n}$. Given a predicate p and data d the value of $R^{v_1,\ldots,v_n}_{x_1,\ldots,x_n}(p)(d)$ is equal to the value of $p(d')$, where d' is obtained from d by assigning to variables $v_1, \ldots, v_n$ values of variables $x_1, \ldots, x_n$ respectively. Note, that renomination composition (primarily in syntactical aspects) is widely used in classical logic, lambda-calculus, and specification languages like Z-notation, B, TLA, etc. The obtained logics are called renomination logics (quantifier-free logics). Their semantics base are predicate algebras of the form $<{}^{V}A \to Bool, \vee, \neg, R^{v_1,\ldots,v_n}_{x_1,\ldots,x_n}>$. Properties of such algebras will specify calculi for renomination logics.

To introduce first-order logics we should specify a new data intension I_{NDQ}. This intension allows an exhaustive search within the class A. It permits to introduce quantification compositions $\exists x$ and $\forall x$ (x is a variable). The value of $\exists x p(d)$ is defined and equal to T, if there is d', differing from d only in the value of x, such that $p(d')$ is defined and equal to T; is defined and equal to F, if for all such d' the value $p(d')$ is defined and equal to F; and is undefined in all other cases. The composition of universal quantification is defined in a similar way. The semantics base of first-order logics are predicate algebras of the form $<{}^{V}A \to Bool, \vee, \neg, R^{v_1,\ldots,v_n}_{x_1,\ldots,x_n}, \exists x>$.

To preserve properties of classical first-order logic we should restrict the class ${}^{V}A \to Bool$ of quasi-ary predicates. This restriction stems from the fact that predicates of first-order logic, being defined on some data, are also defined with the same value on all extensions of this data. Such quasi-ary predicates are called *equitone*. We also introduce different variations of equitone predicates such as maxitotal (necessarily defined on maximal data), local-equitone (equitone for finite extensions only), and equicompatible (extensible to equitone predicates). Logics based on equitone and maxitotal equitone predicates are the "closest" generalizations of classical first-order logic that preserve its main properties. These logics are called *neoclassical logics*. For all these logics corresponding calculi were constructed; their *soundness and completeness* were proved. All necessary mathematical details can be found in [17].

The last class of logics considered here are logics over hierarchic nominats with intension I_{NDH}. Corresponding logics will use composite names of the form $x_1.x_2. \ldots .x_n$ as parameters of renomination and quantification compositions. For this case their definitions should be redefined to take into account hierarchic structure of data.

Summing up, we can conclude that the notion of intensionalized data 1) permits to construct new kinds of semantically-based predicate logics oriented on such data (intensionalized logics), and 2) gives possibility to explain origination of classical logics as logics oriented on "black box" data (propositional logics) or on flat nominats (first-order logics).

6 Conclusion

In the paper we tried to advocate an idea that foundations of informatics can be developed using a gnoseology-based approach. This approach specifies methodological, conceptual, and formal levels of foundations. For the methodological level we have described a number of general gnoseological principles and a system of philosophical categories specifying the main features of the approach. For the conceptual level we have proposed to elucidate basic notions of informatics in integrity of their intensional and extensional aspects. To support this idea we have defined the notion of intensionalized data, and have presented its formalization at the mathematical level. Then for such intensionalized data we have constructed basic intensionalized computability theory and several intensionalized logics of partial predicates. Still, these investigations are at the beginning phase, and we plan to continue them in the direction of developing mathematical formalisms for specification of information systems.

References

1. Nikitchenko, N.S.: A Composition-nominative approach to program semantics. Technical Report IT–TR 1998-020, Technical University of Denmark, ISSN 1396–1608 (1998)
2. Hegel, G.W.F.: Hegel's Science of Logic / translated by A.V. Miller; foreword by J. N. Findlay. London: G. Allen & Unwin (1969)
3. Brookes, B.C.: A new paradigm for information science. The Information Scientist, 1, 03, pp. 103–112 (1976)
4. Robinson, L. Karamuftuoglu, M.: The nature of information science: changing models. Information Research, 15, 4, colis717 (2010). Available at http://InformationR. net/ir/15-4/colis717.html
5. Hjorland, B.: Epistemology and the Socio-cognitive Perspective in Information science. Journal of the American Society for Information science and Technology, 53, 4, pp. 257–270 (2002)
6. Bawden, D.: Smoother pebbles and the shoulders of giants: the developing foundations of information science. Journal of Information science 34, 4, pp. 415–426 (2008)
7. Cisek, S.: Information science in the XXI century: the meta-theoretical research. In: Od książki dawnej do biblioteki wirtualnej. Przeobrażenia bibliologii polskiej. Degen, D.; Fedorowicz, M. (eds.), pp. 47–56. Toruń: Wydaw. Naukowe UMK (2009). In Polish.
8. Bourbaki, N.: Theory of Sets. Berlin: Springer-Verlag (2004)

9. Zins, C.: Conceptual approaches for defining data, information and knowledge. Journal of the American Society for Information science and Technology, 58, 4, pp. 479–493 (2007)
10. Formal Theories of Information: From Shannon to Semantic Information Theory and General Concepts of Information. Sommaruga G. (ed): LNCS, vol. 5363, Springer, New York (2009)
11. Ackoff, R. L.: From data to wisdom. Journal of Applied Systems Analysis 15, pp. 3–9 (1989)
12. Hey, J.: The Data, Information, Knowledge, Wisdom Chain: The Metaphorical link, published at Intergovernmental Oceanographic Commission, 2004, 17 pages. Available at http://web.archive.org/web/20070206032947/ioc.unesco.org/oceanteacher/OceanTeacher2/02_InfTchSciCmm/DIKWchain.pdf
13. Nikitchenko, N.S.: Intensional aspects of the notion of program. Problems of Programming, Kiev, 3–4, pp. 5-13 (2001). In Russian.
14. Apostoli, P., Hinnion, R., Kanda, A, Libert, T.: Alternative Set Theories. In: Philosophy of Mathematics, Irvine A.D. (ed.), pp. 461-491, Elsevier (2009)
15. Nikitchenko, N.S.: Abstract Computability of Non-deterministic Programs over Various Data Structures. In: Perspectives of System Information science. LNCS, vol. 2244, pp. 471–484. Berlin: Springer (2001)
16. Backus, J.: Can programming be liberated from the von Neumann style? A functional style and its algebra of programs. Communs. ACM, 21, pp. 613-641(1978)
17. Nikitchenko, M.S., Shkilniak, S.S.: Mathematical logic and theory of algorithms. Publishing house of National Taras Shevchenko Univ. of Kyiv (2008). In Ukrainian.