INTENSIONAL ASPECTS OF MAIN MATHEMATICAL NOTIONS

Nikitchenko M.S.

National Taras Shevchenko University of Kyiv, Ukraine

Modern mathematics is based primarily on the set-theoretic platform. Thus, the notion of *set* can be considered as the main mathematical notion. But this notion 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 [1]. N. Bourbaki in his books aimed to write a 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 mathematics. Therefore numerous attempts were made to construct set theory without extensionally [2]. The need of non-extensional theories of sets and other main mathematical notions is especially visible in informatics, artificial intelligence, knowledge bases, and other disciplines dealing with the notions of information and knowledge. Therefore it seems reasonable to enhance extensional definitions of main mathematical notions with *intensional* component. Here the intension of a notion 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. The *intension/extension* dichotomy was studied primarily in logic, semiotics, and linguistics; we propose to provide more active investigations of this dichotomy in mathematics too. It means that the main mathematical notions, such as *data* and *function*, should be enhanced with extensional components. Investigations in informatics and logic demonstrate that the notion of *composition*, considered as a function/predicate construction mechanism, should be also added to the list of main mathematical notions. In this paper we continue our investigations on intensionality presented in [3, 4].

1. Intensionalized data. Considering mathematical notions in integrity of their intensional and extensional aspects we obtain possibilities to define more first-level notions as basic notions of mathematical formalisms. Therefore we propose to extend set theory to a *theory of intensionalized data*. Such data can be considered as certain objects with prescribed intensions. This idea is similar to the notion of typed data, but the latter is usually understood in the extensional sense while we aim to emphasize intensional features of data.

The main difficulty in constructing theories of intensionalized data is concerned with the definition of data intension. We propose to start with intuitive understanding of intensions, and then to construct their formal explications.

We begin with the most abstract understanding of data as some objects. Objects can be considered as *unstructured* objects (as *wholes* with intension I_W) or as *structured* object (with *parts*, intension I_P). An object with the intension I_W can be regarded as a “black box” (intuitively it means that nothing is “visible”, and therefore nothing is known about the object, intension I_{WB}) or as a “white box” (everything is “visible” and recognizable, intension I_{WW}). An intermediate intension is denoted by I_{WBW} (“black or white box”).

To come to richer intensions we should treat objects as structured (with intension I_P). We start with simple structures: 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 above specification of object structure permits to call it *hard structure*. Thus, we divide intension I_P into two sub-intensions I_{PH} and I_{PS} specifying objects with hard and soft structures respectively.

We continue with I_{PH} classification that is caused by possible relationships between object parts. Such relationships are classified along the line *tight–loose*. Loose relationships mean that parts are not connected with each other (intension I_{PHL}); tight relations mean that parts are connected (intension I_{PHT}). Considering parts as wholes, we can treat them with intensions of black and/or white boxes. Thus, three new intensions stem from this: I_{PHLB}, I_{PHLW}, and I_{PHLBW} .

Objects with intension I_{PHLB} should be regarded as collections of black boxes. Such objects we call *presets*. Collections of white boxes (intension I_{PHLW}) are called *explicit multi-sets*; if repetition of elements is not allowed then we obtain *explicit sets*. Collections with intension I_{PHLBW} contain “black” and “white” elements (*mixed presets*).

Within intension I_{PHT} we distinguish objects which are constructed of white boxes (as names) related to black boxes (as values of names). We propose to call such objects *nominats* and denote a corresponding intension as I_{ND}. It is important to admit that nominats can model the majority of data structures used in informatics [3, 4].

Thus, we propose to introduce in mathematical practice additionally to the notion of set the above specified notions: presets, multi-sets, mixed presets, and nominats as the basic mathematical notions. These notions are enriched with intensional components and are non-extensional. To realize this idea we have to describe properties and operations for these new notions. In this paper we will consider in some detail only the notions of preset and nominat.

2. Presets and their properties. Intuitively, presets can be understood as collections of externally undistinguished objects (elements) which have hidden content.

A good example of preset is a collection of tickets of an instant lottery. The surfaces of tickets should be covered by opaque material making them “black boxes” that hide the content of tickets.

Having this example in mind we can be specify our understanding of presets by the following intuitive properties:

- each element of a preset is some whole;
- elements are separated from one another;
- elements are independent of one another, i.e., close relations between them are absent;

- all elements "are available," i.e., each element can be obtained for processing;
- exhaustive processing of all elements of a preset is possible;
- elements do not vary until it is explicitly mentioned (the law of identity of elements).

Let us admit that these properties are very weak and do not specify membership relation, so, given a preset and an element, it is not possible to say whether this element belongs to the preset. Also the equality relation is not specified. It is also possible to have many equal elements (duplicates) in a preset, thus, extensionality axiom is not valid. These properties of presets have a negative character restricting possibilities for processing of presets. But what is allowed to do with presets?

Analysis of the above formulated properties leads to the conclusion that the following operations are allowed for presets with the intension I_{PHLB}:

- union $\cup$ which given presets *pr1* and *pr2* yields a new preset consisting of elements of *pr1* and *pr2*;
- nondeterministic choice *ch* which given a preset *pr* yields some element *e* of *pr*;
- nondeterministic choice with deletion *chd* which given a preset *pr* yields some element *e* of *pr* and a preset *pr′* without this element;
- cardinality operation *card* which given a preset *pr* yields the number of elements in *pr*.

Let us admit that intersection of presets is not allowed, contrary to set theory.

The above defined operations *conform to the intension* I_{PHLB} (are *preset–conforming* operations). It means that during their execution these operations will not require additional information hidden in "black boxes" thus they use only that information which is prescribed by the intension. Still, the idea of a preset says that elements contain some hidden content; therefore operations working with this content are also required. The most natural of such operations is *open* operation. Given a preset *pr* this operation constructs a multi-set *ms* which consists of "white box" elements that are content of the elements of the initial preset. We use multi-sets here because cardinality of *pr* and *ms* should be the same. It means that duplicates should be preserved. The *open* operation does not conform to the intension I_{PHLB} because it opens "black boxes". Therefore, theory of presets should contain two parts: one part describes operations that conform the intension I_{PHLB} while the other part specifies more powerful operations which can change intensions of preset elements.

All these considerations about presets still lack formality. The difficult question is what formal model we should choose to support developing of preset theory. We propose to use naturalization approach [4, 5] to construct formal models of finite presets.

The idea of this approach is simple. Let *B* be a class of elements and *Pres*(*B*) be a class of presets with elements from *B*. Class $Nat(B) = Nat \times B^*$ is called a class of *natural data*. These data consist of pairs of the form (k,<b_1, ..., b_m>), where k is a natural number called the *preset cardinality* and the sequence <b_1, ..., b_m> of elements from B is called the *preset base*. Multi-valued (nondeterministic) total injective mapping *nat*: $Pres(B) \rightarrow Nat(B)$ is called *naturalization* mapping; inverse (partial single-valued surjectuve) mapping is called *denaturalization* mapping. Introduction of naturalization mapping is a crucial moment for defining operations that conform to the preset intension. This mapping can be regarded as a formalization of preset intension; and this enables us to reduce an intuitive notion of operations over $Pres(B)$ to formally defined operations over $Nat(B)$. For the latter operations only processing with preset cardinality and projections over preset base are allowed (see details in [4]). Obtained formal class of operation is called a class of *preset–conforming* operations.

Having this definition we can prove that union, choice, choice with deletion, and cardinality operations are preset–conforming. Actually, using such techniques we can formally describe all preset–conforming operations of different types. For example, any preset–conforming operation *op* of type $Pres(B) \rightarrow Nat$ can be presented as a composition of a certain function *na*: $Nat \rightarrow Nat$ and a cardinality operation *card*, thus, $op = na \circ card$. We can also prove, say, that intersection of two presets is not preset–conforming operation.

Summing up, we can say that proposed naturalization approach permits to define all preset–conforming operations (at least for finite presets), thus giving possibility for further developing of preset theory. This theory is weaker than theory of multi-sets and sets because a membership relation is not allowed. At the end of this section we will briefly discuss other approaches to construct theories without extensionality that are weaker than set theory.

We start with Bishop's proposal [6]. Probably, he was the first who introduced the term "preset". Toby Bartels explains that for Bishop a *preset* is like a set without an equality relation; conversely, a set is a preset equipped with an equality relation (http://ncatlab.org/tobybartels/show/preset). This understanding stems from Bishop's three steps definition of a *set*: you should first state how to construct an element of the set; then you should describe how to prove that two elements are equal; and at last you should prove that this (equality) relation is reflexive, symmetric, and transitive. If you only do the first step, then you don't have a set, according to Bishop; you only have a *preset*. A given preset may define many different sets, depending on the equality relation. From this follows that a membership relation is defined for Bishop's presets, but extensionality axiom fails. Thus, our understanding of presets is weaker and different from Bishop's treatment.

Another interesting alternative of traditional set theory is theory of partial sets [7]. In this theory the characteristic function f_S of a set S is considered to be partial. In this case the resulting theory is not extensional. A variant of a partial set theory based on protosets was developed in [8]. A protoset is like a well-founded set except that it has some kind of packaging which hide some of its elements. Again, theories of partial sets are stronger than the theory of presets.

A new line of defining non-extensional theories of sets was developed within category theory (discussion on the topic can be found in [9, 10]).

Such numerous examples of non-extensional set theories give good evidence that many scientists are aware of restrictedness of traditional set theory and importance of intensional set theories; thus, our approach is not an isolated attempt, but is one of representatives of sufficiently elaborated topic. Still, theory of presets is weak to construct rich models used in mathematics and informatics; therefore more powerful structures (notions) are required.

One of such notions is the notion of nominat.

3. Nominats and their properties. Intuitively, a nominat can be considered as a concretization of a preset in which each element consists of "white box" and "black box". To make this abstract consideration more concrete we should involve practical observations which permit to say that white box is a *name* of the black box; and their relation is a *naming* (*nominative*) relation. We call such objects *nominats* and denote a 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*.

A good example of a nominat is a collection of sealed envelopes with addresses written on them. The address is a name (white box) and the content (black box) is the value of a name.

Nominats have the dual nature: first, they may be considered as certain collections of elements; second, they may be considered as functions due to relation that connects names and their values.

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 and mathematics [3]. 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 $1, \ldots, m$ 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.

The main operations over nominats are the following:

- *naming* $\Rightarrow v$ (with name $v \in V$ as a parameter) which given a value a yields a nominat $[v \mapsto a]$;
- *denaming* $v\Rightarrow$ (partial multi-valued operation with name $v \in V$ as a parameter) which given a nominat d yields a value of v in d if it exists;

- *checking* $v!$ (with name $v \in V$ as a parameter) which given a nominat d yields d if the value of v exists in d; or yields $\varnothing_n$ if such a value does not exist;
- *overwriting* ∇ which given two nominats d_1 and d_2 yields a new nominat d consisting of named values from d_2 and those from d_1 which names do not occur in d_2.

Using the naturalization approach described earlier we can prove that these operations *conform to the intension* I_{ND} (are *nominat–conforming* operations). Thus, these operations are allowed for nominats processing. In [5] several theorems were proved that may be considered as descriptions of complete classes of functions over various kinds of intensionalized data, and nominats, in particular.

Now we will describe briefly the distinctions between the notions of set and nominat. We start with the notion of ordered pair (a, b) that can be defined as nominat $[1 \mapsto a, 2 \mapsto b]$ where 1 and 2 are standard names. The notion of ordered pair in set theory has many definitions:

- $(a, b)=\{\{\{a\}, \varnothing\},\{\{b\}\}\}$ – Norbert Wiener, 1914;
- $(a, b)=\{\{a,1\},\{b,2\}\}$ – Felix Hausdorff, 1914 (1 and 2 are two distinct objects different from a and b);
- $(a, b)=\{\{a\},\{a,b\}\}$ – Kuratovski, 1921;
- etc.

It seems that these definitions look not very adequate to the intuitive notion of ordered pair, because they require detailed analysis of bracket structure (Wiener's definition), or are restrictive (Hausdorff's definition), or collapse to singleton $\{\{a\}\}$ when $a=b$ (Kuratovski's definition). It is interesting to admit that in *Principia Mathematica* the notion of ordered pair was considered as primitive, and even N. Bourbaki took the same position. So, introduction of special primitives like ordered pairs (and nominats in our case) is not a new idea.

Concerning further relationships of ordered pairs and tuples with nominats, we would like to emphasize that nominats are more adequate to mathematical practice than tuples. To make this statement more understandable, let us consider questions of operating with tuples and nominats. Indeed, given two tuples $(a_1, \ldots, a_m)$ and $(b_1, ..., b_n)$ we can combine them practically only as concatenation $(a_1, \ldots, a_m, b_1, ..., b_n)$ or $(b_1, ..., b_n, a_1, \ldots, a_m)$. But concatenation is a coarse operation that ignores possible coincidence of some values from $\{a_1, \ldots, a_m, b_1, ..., b_n\}$ representing the same attributes. Thus, a mathematician is forced to make finer combinations of $(a_1, \ldots, a_m)$ and $(b_1, ..., b_n)$ manually, that complicates processing of such data. Instead of this data structure (tuples) we propose to consider nominats. In this case we have more natural combining operations, for example, given nominats $[x \mapsto 9,\ y \mapsto 3,\ z \mapsto 7]$ and $[t \mapsto 9,\ u \mapsto 3,\ x \mapsto 7]$ we obtain $[y \mapsto 3,\ z \mapsto 7,\ t \mapsto 9,\ u \mapsto 3,\ x \mapsto 7]$ as their overwriting combination (cf. with combination of tuples (9, 3, 7) and (9, 3, 7). Also, other combining operations can be defined. This richness of combining operations simplifies processing of nominats compared with tuples. But what is the cause of this richness and simplicity? The answer lies in adequate choice of abstraction level (that is represented by intension) for nominats: the tuple structure is on a much lower level of abstraction than nominat structure, and this impacts both on the operations used to manipulate these data and on the combining forms (compositions) of such functions. For tuple structures, components of data are manipulated by indicating the position (natural number) of the component in the object, and since object transformations often change the position of the components, these changes must be controlled explicitly by the permanent monitoring. As to component names, they are usually stable. Therefore, the abstraction level of "position" is lower than that of "name" since it depends more strongly on other positions than a name depends upon other names. Thus, operating with names (with nominats) is more "soft" with respect to data transformations. The above considerations shortly argue in favour of using nominats as one more basic data structure in mathematics.

Having declared nominats to be more adequate than tuples, we turn now our attention to comparison of classes of functions over nominats and tuples. Traditionally, in mathematics n–ary functions (functions over tuples) are considered as the main class of functions. But does it indeed correspond to mathematical practice?

Let us consider two formulas ($x<y$) and ($y<z$). If we formalise them as binary predicates of type, say, $Int\times Int\rightarrow Bool$, then it will be not easy to formalize formula $(x<y)\&(y<z)$ because this formula specifies ternary predicate constructed from two binary predicates. To define such formalization we should make identical the second argument in the first predicate and the first argument in the second predicate. Thus, operating with n–ary predicates is not easy. Instead, we propose to formalize those formulas as predicates defined over nominats. Such mappings we call *quasiary*. In this case combinations of quasiary predicates is much easier than of n–ary predicates because you should not make transformations of predicate domains.

Let us admit, that the notion of quasiary predicate is implicitly used in traditional mathematics. For example, this notion can be easily "extracted" from Tarski's definition of first-order language (FOL) semantics [11]. This semantics is based on the notion of interpretation which consists of two parts: 1) interpretation of predicate and functional symbols in some structure, and 2) interpretation of individual variables in the domain of this structure. The latter are usually called variable assignments (or evaluations) and can be represented by total mappings (total assignments) from a set of individual variables (names) V into some class of basic values A. In applications partial assignments are often used instead of total assignments. We see that such partial mappings can be treated as nominats. Their class will be denoted by ${}^{V}A$. In this case predicate and functional symbols can be interpreted as partial functions from a class of nominats to *Bool* and A respectively. Such mappings are quasiary predicates and quasiary functions. Their classes are denoted by Pr^{A} and Fn^{A}. In this case formulas and terms of FOL can be interpreted as quasiary predicates and quasiary functions respectively; propositional connectives and quantifiers are treated as operations (we call them compositions) over predicates; substitution of functions into a function or a predicate can be semantically represented by a special operation called superposition composition. This means that two-sorted algebras (with sets of quasiary predicates and functions as sorts and above-mentioned compositions as operations) form a semantic base for FOL. These considerations permit to define and study different logics based on the above type algebras of quasiary predicates and functions.

So, the notion of quasiary mapping, supplied with corresponding intensions of its domain and range, should be also considered as one of the main mathematical notions.

At last, we would like to say a few words concerning intensional aspects of compositions of functions over intensionalized data. In traditional mathematical considerations the role of composition is usually underestimated. But when we look at compositions as fundamental means of system construction, then we will clear see a tight connection of compositions with intensions of data and function. For example, such propositional composition as disjunction, conjunction, and negation conform with treating of predicate domains as presets; quantifiers are oriented on predicate domains as nominats [12]. The same situation can be seen in programming. The simplest compositions are multiplication (sequential execution), branching (the conditional operator *if_then_else*), conditional iteration * (the loop *while_do*), binary overwriting composition [5]. Again, the first three compositions conform with preset intension of domain of functions, and the overwriting composition conforms with nominat intension.

Summing up, we can conclude that the developed notions of intensionalized data and function, which conforms to data intensions, can represent data and function structures used in mathematics and informatics, and besides, such representations look richer and more adequate than traditional set-theoretic representations.

4. Conclusions. Set theory is the main mathematical system that is used for construction of problem domain models. Being well-developed and studied, it gives a nice mathematical instrument for investigations of models constructed on the set-theoretic platform. But at the same time more and more examples demonstrate that in certain cases set theory is not adequate to problem domain formalization especially when only partial information about domain is accessible. The reason of this inadequacy lies in the fundamentals of set theory: membership relation and extensionality principle. For problem domains with incomplete information a membership relation cannot be defined, also the extensionality principle fails. We propose to consider a weaker "set" theory with explicit intensional component. Such a theory may be called theory of intensionalized data. The first-level notions of this theory are notions of preset, set, and nominat. Presets may be considered as collections of "black boxes", sets as collections of "white boxes", and nominats as collections of "grey boxes" in which "white boxes" are names and "black boxes" are their values. In the paper we have defined these notions and described their main properties. In the forthcoming papers we plan to demonstrate how these notions can be used for specifying intensionalized semantics of first-order predicate logics.

REFERENCES

1. Bourbaki N. Theory of Sets. – Berlin: Springer–Verlag, 2004. – 414 p.
2. Apostoli P., Hinnion R., Kanda A, Libert T. Alternative set theories. In: Philosophy of Mathematics: Irvine A.D. (ed.). – Elsevier, 2009. – P. 461–491.
3. Nikitchenko N.S. Intensional aspects of the notion of program // Problems of Programming. – 2001. – N 3–4. – P. 5–13. (in Russian)
4. Nikitchenko M.S. Gnoseology–Based Approach to Foundations of Informatics. // Ermolayev, V. et al. (eds.) Proc. 7-th Int. Conf. ICTERI 2011, Kherson, Ukraine. Vol. 716. – P. 27–40. http://ceur-ws.org/Vol-716/ICTERI-2011-CEUR-WS-paper-1-p-27-40.pdf
5. Nikitchenko N.S. Abstract computability of non-deterministic programs over various data structures // Perspectives of System Informatics. LNCS, vol. 2244. – Berlin: Springer, 2001. – P. 471–484.
6. Bishop E. Foundations of Constructive Analysis. – New York: McGraw–Hill, 1967.
7. Gilmore P. C. The consistency of partial set theory without extensionality // Axiomatic Set Theory. /Jech, Th. (ed.). – American Mathematical Society, 1974. – P. 147–153.
8. Mislove M. W., Moss L., Oles F.J. Partial Sets // Cooper, R., Mukai, K. and Perry, J. (eds.). Situation Theory and Its Applications. – Stanford, 1990. – P. 117–131.
9. Marquis J.-P. Categories, Sets and the Nature of Mathematical Entities. // Benthem J., Heinzmann G., Rebuschi M., Visser H. (eds.) The Age of Alternative Logics: Assessing Philosophy of Logic and Mathematics Today. Logic, Epistemology, and the Unity of Science. – Dordrecht: Springer, 2006. – P. 181–192.
10. Blass A., Gurevich Yu.: Why Sets? (Column: Logic in Computer Science). – Bulletin of the EATCS. – 2004. – N 84. – P. 139–156.
11. Handbook of Logic in Computer Science. Abramsky S., Gabbay Dov M., Maibaum T. S. E. (eds.), in 5 volumes. – Oxford Univ. Press, Oxford, 1993–2001.
12. Nikitchenko M.S., Shkilniak S.S. Mathematical logic and theory of algorithms. – Publishing house of National Taras Shevchenko Univ. of Kyiv, 2008. – 528 p. (in Ukrainian)