Taras Shevchenko National University of Kyiv

1ST WORLD LOGIC DAY

JANUARY 14, 2019

Logic and its Applications

The workshop

Book of Abstracts

Kyiv

Logical Theory for Cyber-Physical Systems: Current State and Outlook

Sergiy Bogomolov
The Australian National University
bogom.s@gmail.com

An Algebraic Explication of Data Structures

Ioachim Drugus
Institute of Mathematics and Computer Science, Chisinau, Moldova
ioachim.drugus@math.md

Data structures are treated here as "constructed objects", or "objects obtained in result of a process of construction". The approach to data structures developed in current research brings into focus the "facture" of these objects, i.e. the manner how these are built (or "constructed") by applying various "construction operations". Accordingly this aspect is explicated here by "aggregate algebras" earlier introduced by the author.

A generalized Boolean algebra (GBA) is defined here as a lattice G with a least element 0, such that for any g in G, the sub-lattice induced over the segment $[0, g]$ (i.e. the set $\{x \in G \mid 0 \leq x \leq g\}$) is a Boolean algebra. It is proved that the class of so defined GBAs coincides with the class of algebras satisfying the Stone's axioms of GBA. Also, an alternative axiomatization of this class is presented.

The intuition behind GBA adopted here is explained in this section as follows. GBAs are treated as "algebras of quantities measured by comparison (in particular, comparison with an etalon)". Such quantities are results of measuring *parts* of a whole (in particular, with the whole) and should be treated "mereologically". Since such quantities are infinitely divisible, GBAs serving as algebras of quantities cannot reflect the process of construction. Accordingly, GBAs are treated here as "carriers" or "supports" of data structures.

An *aggregate algebra A* is a GBA, equipped with two operations called here "construction operations", the two symbols of which are a "superfix" unary operator "^{o}", and an "infix" binary operator "$\circ$", both "invertible" in the sense that the universal closures of next formulas hold:

$$x^{o} = y^{o} \rightarrow x = y,$$

$$x \circ y = x' \circ y' \rightarrow x = x' \,\&\, y = y'.$$

The intuition behind a "construction operation" is that such an operation "keeps the values of arguments within the result of its application", or in other words, that this operation "preserves parthood". This intuition is reflected in the above formulas.

The intuitive treatment by David Lewis of sets as “superstructures over the Matter” presupposes the treatment of sets as (elementary) data structures, and is compliant with the approach to data structures used in current research.

Sources of Contradictions in Zeno’s Paradoxes “Arrow” and “Dichotomy”

Igor Zenonovych Dutsiak
idutsyak@gmail.com

The emergence of paradoxes (contradictions, the source of which is difficult to identify) in the process of cognitive activity is a manifestation of the imperfection of theoretical knowledge. Zeno’s paradoxes became the object of analysis of a huge number of researchers; however, there are no generally accepted explanations of the sources of these contradictions. In this publication, the explanation of sources of contradictions of paradoxes “Arrow” and “Dichotomy” is proposed.

Consider the paradox “Arrow” (assume that the movement exists; in this case, the arrow can fly from one point to another; however, during such a flight at each time moment, the arrow is located in a separate place in space, that is, throughout the entire flight this arrow must remain stationary, and therefore the movement does not exist).

First of all, note that well-known fact that one body can be concerning other body both in the condition of tranquillity, and in the condition of mechanical movement. In this case, it is incorrect to say that during the movement the arrow is stationary in certain places of the trajectory. In fact, only in that place where the arrow stopped its movement, it becomes stationary. Concerning all other places of the trajectory it is more correct to say that the arrow passes them in its movement. It concerns also that place where the arrow began to move independently from the bowstring of the bow, which pushed the arrow.

In this case, there will be a counterargument – like, “which can be movement in the time moments (we designate them by points on a time scale)”? According to this remark, it is important to pay attention to the fact that the concept of a moment is relative. Every moment, actually is very small period, and in smaller scale each point on a time scale will turn into a segment on such scale. At the same time, each period during the body movement corresponds not to place that has the linear dimensions of the body and in which this body is stationary, but that in which the body moves from the beginning of the segment to its end. So, describing the movement of a body, instead of the words “at such time moment the body is in such place of space” it is more correct (and it is consistently) to say “in such small period the body moves in such segment of trajectory”. Formulating the paradox “Arrow” Zeno came into conflict with himself – on the one hand he approves infinite

divisibility of time and space, on the other hand – uses indivisible time moments (adoption of divisibility of the moments will turn them into the periods during which there is a movement, and in this case, the paradox will not emerge).

Consider the paradox “Dichotomy” (assume that the movement exists; in this case, the arrow can fly from one point to another; however, before the arrow flies all distance, it is necessary for it to fly half of this distance; before the arrow flies this half, it is necessary for it to fly half from it, that is, a quarter; since such a condition has to continue indefinitely reducing the distance in half, the arrow cannot start moving, and therefore the movement does not exist). In this paradox, the contradiction is the result of other “manipulations”.

Construct a model of the movement. Replace the passage of the same arrow of different parts of the trajectory with a line of archers who shoot simultaneously at different distances: the first one sends the arrow at some distance; the second is at half of this distance, the third is at a quarter of the mentioned distance, etc. This model (in the extreme case, in the range of sizes of the macro world) can be implemented materially, that is, we can actually conduct such an experiment (trivial from the point of view of everyday experience). The results of such shooting can be summarized as follows: the possibility that the movement will begin does not depend on distance that the body moves. This empirical generalization is contradicted by the reasoning used in the paradox “Dichotomy” – the possibility that the movement will begin, depends on distance that the body moves forward (indeed, in order to start moving the body must travel the smallest distance, and since this does not exist due to the infinite divisibility of the segment, the movement cannot begin). The same is the source of the controversy in the “Achilles and the Tortoise” paradox.

On the Partial Floyd-Hoare Logic Based on Predicate Complement

Ievgen Ivanov
Taras Shevchenko National University of Kyiv
ivanov.eugen@gmail.com

It is known that an inference system for the classical Floyd-Hoare logic [1, 2, 3] becomes unsound in the case when the pre- and postconditions are allowed to have undefined values (i.e. be specified by partial predicates) [4, 5] and Hoare triples {p}f{q} are understood in the weak sense: “if p(d) is defined and true and f(d) is defined, and q(f(d)) is defined, then q(f(d)) is true”.

In this talk we describe a novel modified Floyd-Hoare logic and the corresponding inference system which is sound in this case. The resulting inference system makes use of the operation of predicate complement which maps a partial

predicate to a partial predicate in such a way that the resulting predicate is defined and true on a given data if and only if the input predicate is undefined on the same data, and the resulting predicate is undefined on a given data, if the input predicate is defined on the same data.

We also discuss potential applications of the mentioned modified Floyd-Hoare logic in high-level verification of software which implements engineering and scientific computing algorithms.

References

1. R. Floyd. Assigning meanings to programs. Mathematical aspects of computer science 19, pp. 19-32, 1967.
2. C.A.R. Hoare. An axiomatic basis for computer programming. Commun. ACM 12(10) pp. 576–580, 1969.
3. K. Apt. Ten years of Hoare's logic: A survey – part I. ACM Trans. Program. Lang. Syst. 3(4), pp. 431–483, 1981.
4. A. Kryvolap, M. Nikitchenko, W. Schreiner. Extending Floyd-Hoare logic for partial pre- and postconditions. In Ermolayev, V., Mayr, H., Nikitchenko, M., Spivakovsky, A., Zholtkevych, G., eds.: Information and Communication Technolog in Education, Research, and Industrial Applications. Volume 412 of Communications in Computer and Information Science. Springer, pp. 355–378, 2013.
5. A. Kornilowicz, A. Kryvolap, M. Nikitchenko, I. Ivanov. An approach to formalization of an extension of Floyd-Hoare logic. In: Proceedings of the 13th International Conference on ICT in Education, Research and Industrial Applications. Integration, Harmonization and Knowledge Transfer, Kyiv, Ukraine, May 15-18, pp. 504–523, 2017.

The Multiplicity of Logical Communities

Yaroslav Kokhan

Institute of Philosophy, National Academy of Sciences of Ukraine, Kyiv

yarkaen@gmail.com

Usually, we presuppose that any given science has a unique professional community. But this is not the case of logic. In logic, we see at least four different professional groups.

The first group consists of *logicians-philosophers* and exists from antiquity time. The modern logical infrastructure is made by philosophers.

The second group—*logicians-abstract mathematicians*—arose in 19th century and made logic a strict science. Currently we can see this group dissolves in other professional communities: partly mathematical, partly philosophical.

The third group that investigates different theories of decision (recursion, λ-conversion, abstract computing machines, verbal algorithms) has separated from the second one and now claims to the status of a separate science—theory of algorithms. Recent time, this field is expanded to the information science in a wide sense. So here we meet *information scientists*.

And the fourth group unexpectedly consists from *theorists of formal grammars*, working in mathematical linguistics. Although the founder of this field Noam Chomsky claims that his theory does not concern to logic, as a matter of fact, formal grammars are just a partial case of Post-Smullyan formal systems with rules that originate from Leśniewski's theory of semantic categories.

Taking into account the above, we can conclude that it is necessary to unite all these disparate groups of logicians into a single scientific community.

Automated Theorem Proving in Kyiv: Historical Notes

Alexander Lyaletski
Taras Shevchenko National University of Kyiv

A historical sketch of fulfilling the Kyiv's investigations in the automated theorem proving field beginning from 1962, the time of starting the investigations, and ending today is given. The main Kyiv's results in this field and researchers who obtained them are listed in the chronological order. At that, some connections with the similar investigations of other researchers are indicated.

Justification Logic

Volodymyr Navrotskyi
Institute of Philosophy, National Academy of Sciences of Ukraine, Kyiv
navrotsk@gmail.com

Logic of justification / rejection is an alternative to assertion / negation logic. The first one is used in the construction of those systems of argumentation which, in addition to the schemes deductive logic, has the schemes of non-deductive logic. Such schemes do not necessarily lead from the true premises to the true conclusion. Yet they can be the reasons for the rational adoption of conclusions. What is the criterion of the sufficiency of such reasons? What is the support of a conclusion, if it is not the truth of its premises?

The specific logic of justification is the logic of reasoning with potentially defeated conclusions (*defeasible logic*). The defeasibility of the conclusions is due in

particular to the fact that some inference schemes provide greater support for conclusions than others. If the inference schemes have different force, then this fact must be taken into account in the formulation of semantic rules expressing the conditions for the adoption of conclusions.

What else to consider? What other factors affect the acceptability of the conclusions? Is it the competence of the logic to answer, for example, the question about the reasons for applying inference schemes?

As an alternative to the logic of the truth values of the sentences, justification logic implements an argumentative approach to logic, in which conclusions of reasoning are accepted or rejected, not simply as the result of the use of deductive schemes, but as a result of the competition of arguments.

Program-oriented Composition-Nominative Logics

Mykola Nikitchenko
Taras Shevchenko National University of Kyiv
nikitchenko@unicyb.kiev.ua
mykola.nikitchenko@gmail.com

In the talk we discuss methodological and mathematical aspects of logics that aim to reason with programs.

We start with the notion of *generalized computable function* that is presented by a *program* in a certain *formal language*. Based directly on such formal program models we develop *program logics* of various abstraction and generality levels. We distinguish three levels of development: 1) *methodological (philosophical)*, 2) *scientific* (oriented on computing), and 3) *mathematical* levels.

The *methodological level* should provide us with general laws of development and with a system of categories that form a skeleton of such development.

At the *scientific level* we follow the general development scheme and make particularization of categories obtaining computing notions such as *user, problem, information, program, data, function, name, composition, description* etc. interconnected with such relations as *adequacy, pragmatics, computability, explicativity, origination, semantics, syntax, denotation,* etc. We use *triads* (*thesis – antithesis – synthesis*) to develop these notions later combined into development *pentads*. These notions are considered in integrity of their *intensional* and *extensional* aspects.

At the *mathematical level* we formalize the above-mentioned notions in integrity of their intensional and extensional aspects paying the main attention to the notions of 1) *data*, 2) *function* (specified by its *applicative* properties), and 3) *composition* (considered as function combining mean). Thus, at this level we aim to

develop the theory of *intensionalized program notions* and *intensionalized logics* based on this theory. The initial fragments of the theory and corresponding logics are described. Let us admit that *conventional set theory* is considered as one component of this intensionalized theory.

Though we aim to develop intensionalized logics, we also study their extensional components which are built according to mathematical traditions. Thus, a number of composition-nominative logics oriented on *partial* and *non-deterministic functions* and *predicates without fixed arity* (*quasiary functions and predicates*) over hierarchical nominative data were defined and investigated; corresponding *calculi* were constructed, their *soundness* and *completeness/incompleteness* were proved.

So, the proposed scheme of logic development at three levels seems to be fruitful and permits to construct a hierarchy of new program-oriented composition-nominative logics.

Semantic Properties of Five-Valued Logics

Mykola Nikitchenko, Olena Shyshatska

In the talk the software-oriented five-valued logics of two levels (propositional five-valued logic and the logic of five-valued quasiary predicates) are proposed and studied. Such logics naturally arise for software systems which work with various types of uncertainties and errors.

We describe objectives for constructing and researching five-valued logics. Examples of five-valued functions and predicates that induce corresponding five-valued logics are presented. In particular, an example is described that naturally gives a five-valued set of truth values *EU*={*T, F, e, u, eu*}, where *T* represents "*true*", *F* represents "*false*", *e* represents "*error*, *exception*", *u* represents "*undefined value*", *eu* represents "*and/or an exceptional situation and insufficient information*". Obtained logic is called *EU*- logic.

Composition-Nominative Specification Languages and Logics for the Object-Oriented Programs

Liudmyla Omelchuk
Taras Shevchenko National University of Kyiv
l.omelchuk@knu.ua

Scope of software application includes various problem domains, in particular, safety critical. One of the steps to solve the problem of fast and efficient design of reliable

software is the use of formal methods for software development. Today one of the absolute leaders in application programming is object-oriented programming (OOP). Thus, in the course of construction of modern languages of program specifications it is necessary to take into account the specificity of object-oriented programming languages.

Among the formal specification languages that are able to specify object-oriented programs, are independent of the development environment and can be used to describe the behavior of the entire system as a whole, we should mention Object-Z [Smith 2000], B [Björner, Henson 2008] and RSL [Björner, Henson 2008]. These languages are based on a traditional set-theoretical approach to program formalization and use Zermelo-Fraenkel axiomatics. The use of such developed formalism in relation to the problem of software development allows one to solve effectively specific application problems. This theory is powerful enough, but at the same time its adequacy to programming (adequate semantics, data structure, and composition) is insufficient. These issues actualize the problem of developing of approaches that could lead to the construction of more adequate formalisms of program specifications. We consider one of such approaches to building axiomatic systems of non-deterministic program specifications [Nikitchenko, Omelchuk, Shkilniak 2006], which is based on the composition-nominative method of refinement of the concept of program [Nikitchenko 1998].

Composition-nominative programming studies the systems at different levels of abstraction – abstract, Boolean and nominative (attribute) levels. Systems of the last level are quite adequate for setting of the models of data structures and programs. Thus, the composition-nominative approach provides a single methodological basis to formalize the concept of program specification with their further specification to programming languages of lower level. Axiomatic theory of nominative data [Omelchuk 2007] is developed in the spirit of the theory of admissible sets (S. Kripke, R. Platek, J. Barwise, Yu. L. Yershov). This theory has a number of advantages with respect to the adequacy of the programming: on the one hand, it is enough powerful to generate computable functions over the different data structures, on the other hand, it is not so restrictive as different versions of constructive logic, but it is not excessively powerful and does not allow, for example, the use of axiom of constructing the set of all subsets (compared with set theory by Zermelo-Frankel). Moreover, this theory uses the basic data corresponding to the methods of constructing data in programming. This can increase the adequacy of setting data structures, functions and compositions used in programming languages, and permits to build the systems of program specifications based on the single conceptual framework. Basic data types of programming languages were specified in [Nikitchenko, Omelchuk, Shkilniak 2006], in addition, the computable functions over nominative data were defined.

Based on the composition-nominative method of refinement of the concept of program [Nikitchenko 1998, Nikitchenko, Omelchuk, Shkilniak 2006], axiomatic systems (logics) of software specifications over the nominative data [Omelchuk 2007] and sequent calculi of the composition-nominative logics over nominative data were constructed.

References

Björner, D., & Henson, M. C. (2008). *Logics of Specification Languages*. EATCS Monograph in Theoretical Computer Science. Hardcover: Springer.

Smith, G. (2000). *The Object-Z Specification Language*. Norwell: Kluwer Academic.

Nikitchenko, N. (1998). *A Composition Nominative Approach to Program Semantics*. Techn. Report IT–TR. Technical University of Denmark, Lyngby.

Nikitchenko, N., Omelchuk, L., & Shkilniak S. (2006). *Formalisms for Specification of Programs over Nominative Data.* Electronic computers and informatics (ECI 2006). Košice, Herl'any, Slovakia, 134-139.

Omelchuk, L. L. (2007). *Aksiomatychni systemy specyfikacij program nad nominatyvnymy danymy [Axiomatic Systems of Specifications of Programs over Nominative Data]*. Candidate's thesis. Kyiv [in Ukrainian].

Teaching of the Course "Mathematical Logic" for Students of Humanities

Nataliia Rusina

Taras Shevchenko National University of Kyiv

rusina@knu.ua

Nowadays, the discipline "Mathematical Logic" should be included in the curriculum of not only natural sciences faculties. Students of the humanitian faculties of universities should have informatics and mathematical competencies, which include: development of the culture of logical and algorithmic thinking; the ability to logically justify a statement [Nikitchenko M.S., Shkilnyak S.S.]. In turn, having an elementary complex of logical concepts will allow students to understand the disciplines such as "Mathematics" and "Computer Science" much better.

The course "Mathematical Logic" for students of the Faculty of Philology of the Taras Shevchenko National University of Kyiv provides for the study the following topics:

- concept of proposition; logical operations; composed propositions;
- formulas of propositional algebras; truth tables; tautologies;
- equivalence of formulas;
- normal forms of logical functions; disjunctive normal forms and conjunctive normal forms;

- logical consequence, based on the propositional algebra; consistency of the set of propositions;

- sequents and sequent forms for propositional logic;

- logic of predicates; quantifiers;

- formulas of predicate logic; equivalent formulas; everywhere true formulas; prenex formulas;

- sequents and sequent forms for predicate logic.

Studying the course "Mathematical Logic" contributes to: intellectual development of the students; development of their logical thinking; memory enhancement; ability to analyze, classify and generalize.

Mastering the skills of logical thinking, permanent use of logical techniques and methods will lead to the formation of mathematical and informational competencies for further study and professional work.

The teaching experience of the discipline "Mathematical Logic" gives grounds to offer introduction of the same course in other educational programs of the humanities.

References

Nikitchenko M.S., Shkilnyak S.S., (2008) Mathematical Logic and Theory of Algorithms: A Textbook. Kyiv [in Ukrainian].

Algebras of General Non-Deterministic Predicates

O. S. Shkilniak, S. S. Shkilniak
Taras Shevchenko National University of Kyiv
sssh@unicyb.kiev.ua

Logics of general nondeterministic quasiary predicates, called *GND*-predicates, are defined and investigated. These logics are program-oriented logical formalisms that reflect such properties of programs as partiality, nondeterminism, and non-fixed arity. *GND*-predicates generalize partial predicates of the relational type. The main attention is paid to the construction of composition algebras of *GND*-predicates. Compositions of *GND*-predicates are described, their properties are formulated. For these predicates, such important laws of traditional logic as the law of absorption and the law of distributivity for $\vee$ and & are not valid. Various types of *GND*-predicates are identified. *GND*-predicates can be modeled as 7-value total deterministic predicates (*TD*7-predicates). A 7-element algebra of truth values of *TD*7-predicates is defined and all of its subalgebras are described. Each such subalgebra induces a corresponding algebra of *TD*7-predicates, which then induces the algebra of *GND*-predicates. This makes possible to identify a number of important composition algebras of general nondeterministic predicates. The

languages of pure first-order logics of *GND*-predicates and their interpretations are described. The relations of a logical *G*-consequence and a logical *G*-equivalence are introduced. The relation of the logical *G*-consequence is monotonic, reflexive, and transitive; for it the properties of the decomposition of formulas are satisfied. On the basis of these properties, it is planned to construct calculi of sequential type for the logic of *GND*-predicates.

Many-Valued Logics in the UML/OCL Model

Olena Shyshatska
Taras Shevchenko National University of Kyiv
shyshatska@knu.ua

From information point of view existing databases contain a large amount of incomplete, undefined, and ambiguous information (data). Such data must be processed in a special way. Consequently, researching and constructing of new program-oriented logics becomes important. The peculiarity of such logics is the use of special truth values (indicating undefined value, errors, etc.).

We consider the evolution of using many-valued logic on the example of the formal language of object constraints (OCL). The context is the UML/OCL model – the object model in the UML notation (class diagram) with integrity constraints using OCL expressions.

OCL is developing by the Object Management Group (OMG) of Computer Standards Consortium. The first official version 2.0 of the language is dated 06/05/2006 [1]. Current version 2.4 is dated 02/03/2014 [2].

We illustrate the problems that appear during modeling a database.

Problem_1. Modeling the situation where the attribute value is not yet known (for example, the email address of a customer is unknown at the time of the first contact, but will be added later) or does not apply to this specific object instance (e.g., the customer does not have an email address).

Problem_2. An invalid value can signal an error in the evaluation of an expression. An example for an expression that is defined by a partial function is the division of integers. The result of a division by zero is undefined.

In UML/OCL (OCL, v.2.0) the result of solving such problems is a special undefined value $\perp$. Each domain (set of values) of a basic type contains this value. This usage of undefined values is well known in database modeling and querying with SQL, in the Extended ER-Model, and in the object specification language TROLL light. The problems with partial functions can be eliminated by including an invalid value $\perp$ into the domains of types. For all operations, we can then extend their interpretation to total functions. The interpretation of operations is considered strict

unless there is an explicit statement in the following. Hence, an invalid or null argument value causes an invalid operation result. This ensures the propagation of error conditions.

In the version 2.3 (2012), each domain of the base type contains two special values: ε (null-value or undefined) and ⊥ (invalid value).

The result of solving *Problem_1* and *Problem_2* is introduction of the values ε and ⊥, respectively. The standard Boolean type, supplemented with special values, defines the set of true values of the three-dimensional (older versions of OCL) and four-dimensional logic (OCL version of 2012) UML/OCL.

We describe operations and their interpretations for basic, collection, structural and object types.

We formally define the syntax and semantics of the expressions of OCL and give an exact definition of the notion of context, invariance, pre-/postconditions. We compare the using of special values in different language specifications.

Three- and four-valued logics of the OCL language are described at the propositional level. We propose to extend the set of special values to the third value. Its necessity is obvious in *Problem_2* for evaluating the expression $x/y^{\omega}\ z$. As a result, domain of type Boolean is five-valued set of values *EU*={*T, F, e, u, eu*}, where *T* represents "true", *F* represents "false", *e* represents "error, exception", *u* represents "undefined value", *eu* represents "and/or an exceptional situation and insufficient information". Obtained logic is called EU- logic [2].

References

1. Object Constraint Language. OMG Available Specification Version 2.0. – 2006. [online] https://www.omg.org/spec/OCL/2.0/PDF

2. Object Constraint Language. OMG Available Specification Version 2.4. – 2014. [online] https://www.omg.org/spec/OCL/2.4/PDF

3. Nikitchenko M., Shyshatska O. Semantic properties of Five-Valued Logics // Problem of Programing. – 2018. – № 1. – P. 22-35.

Logical Foundations of Knowledge Representation and Processing in Logical-Computing Semantic Network

Andrew Yalovets

Institute of Software Systems, National Acedemy of Sciences of Ukraine, Kyiv

yal@isofts.kiev.ua

Logical Time and Category Theory

Grygoriy Zholtkevych, Lyudmyla Polyakova
V. N. Karazin Kharkiv National University, Kharkiv
g.zholtkevych@karazin.ua, l.yu.polyakova@karazin.ua

The concept of time is one of the central concepts of philosophy and science in whole. The trend towards the widespread use of distributed computing, which is being observed in recent years as a technological response to the practical achievement of the upper limit of processors performance and the development of communication tools, has put this philosophical concept in the spotlight of researching distributed, parallel, and concurrent computational systems. In addition, the developing tendency to integrate cybernetic and physical systems, which has been accelerated with developing Internet-of-Things, increased the interest in such research.

In 1978, L. Lamport showed that the metric concept of time leads to a whole complex of contradictions in the simulation of distributed, parallel, and concurrent computing processes. The basis of this complex of contradictions lies in the impossibility of exact synchronisation of timers embedded in different computing devices.

As a solution to the problem, L. Lamport proposed to use the notion of logical time based on the concept of logical clocks whose ticks (models of events) are ordered in accordance with the causality relationships. In some sense, we may say that L. Lamport based his approach on the philosophical conventionalism, whose first proponent was the outstanding mathematician, theoretic physicist, engineer, and philosopher of science Henri Poincaré.

These abstract concepts lead to two approaches to the design of complex distributed systems, namely, synchronous and asynchronous design, which based on the event- and state-based approaches respectively to modelling logical time. It is natural to expect that two pictures based on these two mentioned approaches are similar. In the simplest cases, these pictures are indeed similar in the sense that there is an adjunction between the categories underlying the respective models.

Taking into account the reasoning mentioned above it is natural to determine the following goal: to develop the general theory of logical time using category-theoretic language and methods.

Guided by this goal, the authors obtained a number of preliminary results, namely

(1) the category of clock structures has been defined; this category is used to define event-based models of logical time;

(2) the category of clock structures has been studied; particularly,

– the category-theoretic understanding has been found for some important properties of logical time;

– the special representation of this category objects has been developed;

(3) the subcategory of linear clock structures in the category of clock structures has been defined; this category is used to define physical models of logical time;

(4) the analogue of Szpilrajn Theorem has been proven; this result ensures reducing general clock structure to the corresponding system of linear clock structures;

(5) the category of schedules has been defined; this category is a bridge between event-based and state-based modelling approaches;

(6) equivalence between the categories of linear clock structures and schedules has been proven.

How to Create a Complete Proposional Calculation

V. Zubenko

Taras Shevchenko National University of Kyiv