AISB/IACAP World Congress 2012

Birmingham, UK, 2-6 July 2012

Symposium on the History and Philosophy of Programming

Liesbeth De Mol and Giuseppe Primiero (Editors)

Part of

Foreword from the Congress Chairs

For the Turing year 2012, AISB (The Society for the Study of Artificial Intelligence and Simulation of Behaviour) and IACAP (The International Association for Computing and Philosophy) merged their annual symposia/conferences to form the AISB/IACAP World Congress. The congress took place 2–6 July 2012 at the University of Birmingham, UK.

The Congress was inspired by a desire to honour Alan Turing, and by the broad and deep significance of Turing's work to AI, the philosophical ramifications of computing, and philosophy and computing more generally. The Congress was one of the events forming the Alan Turing Year.

The Congress consisted mainly of a number of collocated Symposia on specific research areas, together with six invited Plenary Talks. All papers other than the Plenaries were given within Symposia. This format is perfect for encouraging new dialogue and collaboration both within and between research areas.

This volume forms the proceedings of one of the component symposia. We are most grateful to the organizers of the Symposium for their hard work in creating it, attracting papers, doing the necessary reviewing, defining an exciting programme for the symposium, and compiling this volume. We also thank them for their flexibility and patience concerning the complex matter of fitting all the symposia and other events into the Congress week.

John Barnden (Computer Science, University of Birmingham)
Programme Co-Chair and AISB Vice-Chair
Anthony Beavers (University of Evansville, Indiana, USA)
Programme Co-Chair and IACAP President
Manfred Kerber (Computer Science, University of Birmingham)
Local Arrangements Chair

Foreword from the Symposium Chairs

Liesbeth De Mol[1] and **Giuseppe Primiero**[2]

Given the significance of computing for modern society, the relevance of its history and philosophy can hardly be overestimated. Both the history and philosophy of computing only started to develop as real disciplines in the '80s and '90s of the previous century, with the foundation of journals (e.g. the IEEE Annals on the History of Computing, Minds and Machines and the like) and associations (SIGCIS, IACAP, . . .), and the organization of conferences and workshops on a regular basis. A historical awareness of the evolution of computing not only helps clarifying the complex structure of the computing sciences, but it also provides an insight in what computing was, is and maybe could be in the future. Philosophy, on the other hand, helps to tackle some of the fundamental problems of computing: the semantic, ontological and functional nature of hardware and software; the relation of programs to proofs and, in another direction, of programs to users; the significance of notions as those of implementation and simulation, and many more. The aim of this conference is to zoom into one fundamental aspect of computing, that is the foundational and the historical problems and developments related to the science of programming.

Alan Turing himself was driven by the fundamental question of 'what are the possible processes which can be carried out in computing a number'. His answer is well-known, and today we understand a program as a rather complex instance of what became known as the Turing Machine. What is less well-known, is that Turing also wrote one of the first programming manuals ever for the Ferranti Mark I, where one feels the symbolic machine hiding on the back of the Manchester hardware. This was only the beginning of a large research area that today involves logicians, programmers and engineers in the design, understanding and realization of programming languages.

That a logico-mathematical-physical object called 'program' is so controversial, even though its very nature is mostly hidden away, is rooted in the range of problems, processes and objects that can be solved, simulated, approximated and generated by way of its execution. Given its widespread impact on our lives, it becomes a responsibility of the philosopher and of the historian to study the science of programming. The historical and philosophical reflection on the science of programming is the main topic at the core of this workshop. Our programme includes contributions in

1. the history of computational systems, hardware and software
2. the foundational issues and paradigms of programming (semantics and syntax, distributed, secure, cloud, functional, object-oriented, etc.).

Our wish is to bring forth to the scientific community a deep understanding and critical view of the problems related to the scientific 'paradigm' represented by the science of programming. The kind of questions analyzed and relevant to our task are:

- What was and is the significance of hardware developments for the development of software (and vice versa)?
- In how far can the analogue and special-purpose machines built before the 40s be understood as programs and what does this mean for our conception of 'program' today?
- How important has been the hands-off vs. the hands-on approach for the development of programming?
- What is the influence of models of computability like Church's lambda-calculus on the development of programming languages?
- Which case studies from the history of programming can tell us today something about future directions?
- Is programming a science or a technology?
- In how far does it make sense to speak about programming paradigms in the sense of Kuhn?
- What are the novel and most interesting approaches to the design of programs?
- How do we understand programs as syntactical-semantical objects?
- What is the nature of the relation between algorithms and programs? What is a program?
- Which problems are the most pressing ones and why are they relevant to more than just programmers?
- How can epistemology profit from the understanding of programs' behavior and structure?
- What legal and socio-economical issues are involved in the creation, patenting or free-distribution of programs?

The invited speakers for this symposium are Gerard Alberts and Julian Rohrhuber. *Gerard Alberts* (University of Amsterdam) is a well-known historian of computing. He is the series editor of the Springer book series on the history of computing and one of the editorial members of the leading journal on the history of computing, viz. IEEE Annals for the history of computing. He is also the project leader of the SOFT-EU project. In his talk, he is going to tackle the rather philosophical question: What does software mean?, by studying how it developed historically. *Julian Rohrhuber* (Robert Schumann School of Music and Media in Düsseldorf) is a co-developer of the open source computer language SuperCollider, a language for real time audio synthesis and algorithmic composition and an experienced live coder. Apart from this more 'practical' work, he has made various contributions to philosophy of science in general and the philosophy of programming in particular. He will present some of his philosophical reflections on programming. As an additional Special Event, Julian can also be seen in action during a live performance on Wednesday July 4th titled "*When was the last time you spent a*

[1] Postdoctoral fellow of the Fund for Scientific Research – Flanders, CLMPS, Ghent University, email: elizabeth.demol@ugent.be
[2] Postdoctoral fellow of the Fund for Scientific Research – Flanders, CLMPS, Ghent University, email: giuseppe.primiero@ugent.be

pleasant evening in a comfortable chair, reading a good program" (Jon Bentley). *Live coding to celebrate the Turing Centennial*, by the Birmingham Ensemble for Electroacoustic Research and Julian Rohrhuber playing improvised algorithmic network music.

The other contributors to this Symposium and their talks are:

- Wolfgang Brand, *Two Approaches to One Task: A Historical Case Study of the Implementation and Deployment of two Software Packages for the Design of Light-Weight Structures in Architecture and Civil Engineering*
- Selmer Bringsjord and Jinrong Li, *On the cognitive science of computer programming in service of two historic challenges*
- Timothy Colburn and Gary Shute, *The Role of Types for Programmers*
- Edgar G. Daylight, *A Compilation of Dutch Computing Styles, 1950s–1960s*
- Vladimir V. Kitov, Valery V. Shilov, Sergey A. Silantiev, *Anatoly Kitov and ALGEM algorithmic language*
- Shintaro Miyazaki, *Algorhytmic listening 1949-1962. Auditory practices of early mainframe computing*
- Pierre Mounier-Kuhn, *Logic and computing in France: A late convergence*
- Allan Olley, *Is plugged programming an Oxymoron?*
- Uri Pincas, *On the Nature of the Relation between Algorithms and Programs*
- Nikolay v. Shilov, *Parallel Programming as a Programming Paradigm*

Our programme will be followed by a double session on *Philosophy of Computer Science meets AI and Law*, organized by Rainhard Bengez (TU München) and Raymond Turner (University of Essex).

The Symposium on History and Philosophy of Programming is intended as a follow-up to the First International Conference on History and Philosophy of Computing (`www.computing-conference.ugent.be`), a IACAP sponsored event. The Conference, which took place in November 2011 at Ghent University, represented a first approach to build a community of philosophers and historians working in the area of the computational sciences. The present smaller Symposium will be a bridge to the Second edition of the History and Philosophy of Computing Conference, to be held in October 2013 in Paris. We hope everyone interested in the historical and systematic study of computational sciences and their intersections with other sciences and its applications will get involved in what promises to be a crucial and exciting research area.

Anatoly Kitov and ALGEM algorithmic language

Vladimir V. Kitov[1] and **Valery V. Shilov**[2] and **Sergey A. Silantiev** [3]

Except several publications (see, for example [4]), many achievements of Soviet programmers for the period from 1950 till 1980 practically are still remained unknown abroad. The reasons for this are several secrecy of some works, many open papers were published in Russian language and thus were unavailable for foreign scientists etc. But these achievements were very considerable. It is sufficient only to mention such original developments as REFAL metalanguage for formal language text processing (V. F. Turchin, middle of 1960) or El'-76 high level language (V. M. Pentkovsky, middle of 1970) which was Assembler language as well.

One of the scientists who made significant contribution to the theory and practice of algorithmic languages development was Anatoly Kitov (1920-2005) – outstanding Russian scientist in the field of informatics and computing (Fig. 1) . Another famous Russian scientist, IEEE Computer Society Computer Pioneer academician Alexey Lyapunov called Anatoly Kitov the first knight of Soviet cybernetics. This was not accidentally. Anatoly Kitov was the real pioneer and the words the first and for the first time can be applied to all stages of his scientific career. A. Kitov was the author of the first in the USSR positive article about cybernetics which was not recognized by official Soviet communist ideology. He had published the first Ph. D. thesis on programming, the first Soviet book about computers and programming, the first articles on non-arithmetic utilization of computers. He was the author of the first project of wide-national computer network, the first national textbook on computer science, the first scientific report on management information systems (MIS), etc. He designed the most powerful Soviet computer of that time, established the first scientific computer Centre (the so called Computer Centre nr 1 of the USSR Ministry of Defense), developed the associative programming theory, created the standard industrial management information system (for the Ministry of Radio Engineering Industry) etc. The total amount and innovative quality of his scientific works are really impressive. Unfortunately due to some political reasons his research activity was not officially recognized in Soviet Union [6].

Figure 1. Anatoly Kitov, c1965

In the second half of nineteen fifties Anatoly Kitov for the first time formulated proposals for complex automation of information processing and state administrative management on the base of Integrated computer centre state network (ICCSN). On January 7, 1959 A. Kitov sent in the Central Committee of the Communist Party of the Soviet Union the letter about the necessity of National economy automated management system on the base of ICCSN. It was first in the world proposal on designing of national state automated system for economics management. The leadership of USSR partly adopted Kitov's project but the main idea about the structural reconstruction of National economy management system was rejected [7].

That is why in Autumn1959 Anatoly Kitov sent the second letter in the Central Committee in which he proposed the new innovative project which advanced the modern times on several dozen years. It was the project named Red Book establishing of integrated computer centre state network of dual designation (for economics management and defense control). But once more due to the political reasons this project was rejected and, moreover, its author was excluded from the Communist Party, dismissed from his job at Computer Centre and later discharged from the Soviet Army.

At the beginning of nineteen sixties A. Kitov was the head and scientific supervisor of the group of programmers who were developing large program complex for real time military computer used in air defense system. At this time he was working at Scientific Research Institute nr 5 of the Ministry of Defense. Material which determined the specifications for the future programming language had been got by A. Kitov from his practical experience during the realization of above mentioned project. However he had begun development of ALGEM (ALGorithms for Economy and Mathematics) language some years later, when after his dismissal from the army he worked at Main Computer Centre of the State Radio-Electronic Committee. ALGEM was designated for the programming of the economical, mathematical, logical and control (including the real time control of the technical systems) tasks. In particular the extremely important was the aim of this work to design the language for the programs for processing the large (super large at that time) information arrays of complex but determined structure. In 1965 the first version of ALGEM was finally developed. The expertise of ALGEM was fulfilled in some Soviet scientific institutions and in particular at Computer Centre of Siberia Branch of USSR Academy of Science (facsimile of the letter from the USSR State Committee for Scientific Research Coordination to Siberia Computer Centre with the request of expertise is shown on Fig. 2).

In 1967 A. Kitov published the monograph The programming of informational logical tasks in which for the first time ALGEM lan-

[1] Institute of History of Natural Sciences and Technics Russian Academy of Science, email: vladimir.kitov@mail.ru
[2] MATI – Russian State Technology University, email: shilov@mati.ru
[3] MATI – Russian State Technology University, email: intdep@mati.ru

Figure 2. The letter of USSR State Committee for Scientific Research Coordination.

guage was described [1]. This language was realized in the system of computer programming ALGEM ST-3 (ST-3 – Syntax-directed Translator, the 3rd version) described in 1970 in monograph [5]. Besides ALGEM the system included translator and standard subroutines library. The system ALGEM ST-3 was realized on the base of the second generation computer "Minsk-22" (Fig. 3).

Figure 3. Soviet computer Minsk-22

The development of this computer was ended in 1964. For that time it was middle class computer with the efficiency of 5-6 thousand operations per second, ferrite core main memory with the 8196 cell capacity of 37 bit each and external memory on magnetic tape with the 1.6 million cells. Computer was produced by series from 1965 till 1970. Minsk-22 was one of the most mass computers for that time and altogether 953 machines were manufactured. It was installed at hundreds of computer centres in various Soviet ministries and later in some socialist countries. Appropriately ALGEM ST-3 included in Minsk-22 software was also had several hundred of installations and was widely used during the development of various applied systems which were designated for the processing of hospital charts, application forms, results of experiments etc.

According to the concept of the author ALGEM must have been the procedure-oriented programming language. That is why Algol-60 was selected as a base for the new language. But the practical orientation of ALGEM determined the deep modification of the basic language and introduction of serious alterations: new block nesting mechanism, new variable types and also the special advanced instruments for the work with the values densely packed in computer memory cells.

Program structure is the same as in Algol: declarations followed by operators in order of their execution. Variables are localized in the block: *begin end*. The program can be only the block. Four variable types are determined in ALGEM: *integer, real, Boolean, string*. Values of numerical variables may be numbers, values of logical variables logic values and of string variables strings in contrast to Algol.

In ALGEM there is a possibility to describe the structure of variables by means of declarator shape (in Russian *vid*). It points the quantity and type of symbols contained in variable value:

integer P *shap*e 9 (5) means that integer variable P has the length of five decimal digits;
integer P *shape* 7 (3) P has the length of three octal digits;
integer P *shape* 1 (8) P has the length of eight binary digits;
string date *shape* 99 – L(10) – 9(4) means that string variable *date* has the following structure two decimal digits, space, up to 10 letters, space, four decimal digits (for example 21 September 2012).

In declaration of real variables you may point the sign of number and exponent, location of decimal point and possibility of rounding.

These possibilities were introduces in the language because of necessity for programming the air defense tasks which demanded the highest possible compact value package in memory cells.

Variables in ALGEM language may be simple and composite (composite variables have not *shape* tag). Composite variable contains other variables including composite as well and in fact it is a record type.

Composite arrays are also provided in the language. They are formed from composite variables of similar structure. Declaration of composite value is always ended by symbol *level*. Thus, there is the opportunity to work with arrays of records (i. e. tables). All these means make it possible for ALGEM to solve specific economical and management tasks of any level (they are also attributable to the systems of real time, in particular to air defense systems).

There are also compound operators in the language except blocks. It provides unlimited nesting of blocks and composite operators as well. The operators are the same as in Algol: assignment operator, go to operator, conditional jump, loop operator. Instead of procedure operator it was introduced procedure-code operator (that is why there is no declaration of procedures among list of declarations).

Not consider further details of the differences between two languages we only want to mention that ALGEM mainly was designed for practical purposes of industrial programming while Algol is a classical language for algorithms presenting.

ALGEM also included the best instruments of the functional programming languages Lisp and IPL-V but it was the essential extension of these languages by addition of new list structures and procedures for their processing. For example, Lisp functions provide the processing of two adjacent elements of linear and chain lists. Kitov introduced generalized list structures node lists and nested lists. For the declaration of list variables it was introduced list declarator. Together with the *level* symbol it forms the pair of parentheses. List declarations begins with *list* symbol followed by declaration of list header and then by list element structure. The list element may be the list itself.

One more A. Kitov's monograph "The programming of economical and management tasks" [2] was issued in 1970. Next year this book was translated in German language [3] (covers of Kitov's four monographs about ALGEM language and it implementation are

shown on Fig. 4). New version of the language ALGEM-2 was presented in this book. This language was further development of ALGEM and was also oriented on the solving of economical and management tasks in information retrieval systems.

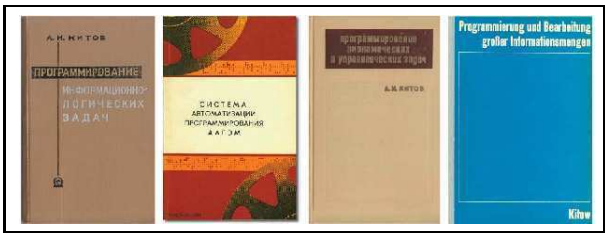

Figure 4. Anatoly Kitov's books on ALGEM languag

ALGEM algorithmic language gained wide popularity and many programs were written on it. For example, information retrieval system "Setka-5" designed for searching the documents upon requests was programmed on this language. This system was realized on Minsk-22 computer and could process document arrays up to 120,000 items stored on 16 magnetic tapes. Search time in the array of 10,000 items (i. e. on one magnetic tape) was 4-5 minutes.

In whole it may be said that ALGEM language was very successful product oriented on industrial programming. It combined the best features of procedure-oriented languages (Algol-60), languages for complex data structures processing (COBOL) and list-processing languages (Lisp, IPL-V). ALGEM realized principles of associative programming developed by A. Kitov. In result ALGEM became one of the most popular programming languages in USSR in 1970-s.

Later A. Kitov developed and introduced one more original programming language NORMIN designed for solving the tasks in medicine sphere but this is the theme of the separate investigation.

Today one can say that programming has not only osmotically infused scientific and artistic research alike, but also that those new contexts elucidate what it may mean to be an algorithm. This talk will focus on the 'impatient practices of experimental programming, which can never wait till the end, and for which it is essential that the modification of the program in some way integrates with its unfolding in time. A contemporary example is *live coding*, which performs programming (usually of sound and visuals) as a form of improvisation.

Early in the history of computer languages, there was already a need felt for reprogramming processes at runtime. Nevertheless, this idea was of limited influence, maybe because, with increasing computational power, the fascination with interactive *programs* eclipsed the desire for interactive *programming*. This may not be an accidental omission, its reasons may also lie in a rather fundamental difficulty, on which we will focus here.

In itself, the situation is almost obvious: not every part of the program-as-description has an equivalent in the program-as-process. Despite each computational process having a dynamic nature, an integration of programming into the program itself must in principle remain incomplete. As a result, a programmer is forced to oscillate between mutually exclusive perspectives. Arguably, this oscillation reveals a specific internal contradiction within algorithms, a necessary obstacle to semantic transparency. By calling this obstacle *algorithmic complementarity*, I intend to open it up for a discussion in a broader conceptual context, linking it with corresponding ideas from philosophy and physics.

Here a few words about this terminology. Complementarity has been an influential idea in conceptualising the relation between the object of investigation, as opposed to the epistemic apparatus and the history of practice. Originating in the psychology of William James, where it referred to a subjective split of mutually exclusive observations, Niels Bohr used it to denote the existence of incommensurable observables of a quantum system (position vs. momentum, time vs. energy). Independent of the particular answer Bohr gave, complementarity raises the question of whether such a coexistence is induced by the experimental system or already present in the subject matter observed. Accordingly, in the case of programs, we may ask whether this obstacle is essential to their nature or whether it is a mere artefact of a specific formalisation. Algorithms, arguably situated between technical method and mathematical object, make an interesting candidate for a reconsideration of this discourse.

The existence of an obstacle to semantic transparency within algorithms and their respective programs need not mean a relative impoverishment of computation. Conversely, prediction is the wager and vital tension in every experimental system, as well as in interactive programming. After the conceptual discussion, I will try to exemplify this claim by introducing a few examples in the recent history of *live coding*. Again and again surfacing in form of symptoms such as an impossibility of immediacy, I hope this practice will be conceivable in terms of having algorithmic complementarity as one of its driving forces.

REFERENCES

[1] Kitov A. I. programmirovanie informacionno-logicheskih zadach. ?oskva: Sovetskoye radio (programming for information-logical problems, in russian), 1967.

[2] Kitov A. I. programmirovanie ekonomicheskih i upravlencheskih zadach (programming for economic and management problems, in russian), 1971.

[3] Kitov A., *I. Programmierung und Bearbeitung Grosser Informationsmengen*, B. G. Teubner Verlagsgesellschaft, 1972.

[4] Shura-Bura M Ershov A., *The Early Development of Programming in the USSR. In A History of Computing in the Twentieth Century*, Academic Press, 1980.

[5] Borodulina N. G. et al., *Sistema avtomatizacii programmirovaniya ALGEM (ALGEM: automation of programming system, in Russian)*, Statistika, 1970.

[6] Shilov V. V. Kitov V. A., *Anatoly Kitov ? Pioneer of Russian Informatics*, History of computing. Learning from the past, Springer, 2010.

[7] Shilov V. V. Kuteinikov A. V., 'Asu dlya sssr: pis?mo a. i. kitova n. s. khrushchevu, 1959 g. (automated management systems for the soviet union: A 1959 letter from a. i. kitov to n. s. khrushchev, in russian)', *Problems of the history of science and technology*, (3), 45–52, (2011).