Bulletin of the **S**ection of **L**ogic
Volume 8/4 (1979), pp. 206–211

reedition 2010 [original edition, pp. 206–213]

COLLOQUIUM ON LOGIC IN PROGRAMMING 10-15 September 1978, Salgótarján (Hungary)

Nowadays computer science is one of the most important fields where mathematical logic is applied. That is the reason why the Hungarian mathematical society, named after János Bolyai, organized a colloquium on mathematical logic in programming. The colloquium took place in a picturesque small Hungarian town, Salgótarján, in the second week of September of 1978, directly after the Mathematical Foundation of Computer Science (MFCS '78, Zakopane).

In this review we do not intend to give a detailed report on the colloquium, but to use this occasion to discuss the "state of art" of the mathematical logical approach toward programming theory.

The basic problem of the mathematical foundation of computer science is the one of the correct, formal semantics of programming languages. This theoretical question is coupled with a practical one: giving methods for proving properties of programs. These demands make natural the use of mathematical logic: semantics and proof procedures have been studied mainly in this field.

Before we start discussing what mathematical logic can give to programming, we should say something about *algebraic semantics*, since this is closely related to logic. Every kind of semantics that meets Frege's principles ("meaning of some compound expression is obtained unconditionally as a combination of the meanings of its constituent parts") can be defined naturally in an algebraic way. This applies also to programming languages, as Z. Markusz and M. Szóts' talk showed, when Montague's Universal Grammar is used. Algebraic semantics is widely studied, it is enough to refer to the activity of the ADJ group (E. S. Wagner, J. B. Wright, J. A. Goguen and J. W. Thatcher). During the colloquium there were some talks based on algebraic tools. The most important of them is the one held by *B. Courcelle*, analysing recursive program schemes.

Semantics based on mathematical logic is not independent of algebraic semantics since the languages of logic themselves can be treated in an algebraic fashion (algebraic logic). The connection between these two kinds of semantics was discussed at the colloquium by I. Németi. He showed how algebraic semantics can be derived from model theoretic semantics. He gave a formalised version of Frege's principle: for a given language (that is for a given triple

<set of expressions, class of models, meaning function>)

a grammar is adequate, if the kernel of the meaning function is a congruence relation with respect to the rules of the grammar. The connection between algebraic and logical semantics is not only a theoretical one. As the language of algebra (the language of equations) can be considered as a sublanguage of the classical first order logic, they can be used together. *A. Bertoni, G. C. Mauri* and *P. A. Miglfoli* used this phenomenon describing abstract data types.

One of the important questions of logical semantics is how to choose an appropriate *language* for describing programs. Though, the most widespread and most commonly known language of logic is the predicate calculus, there are several attempts to use higher order or non classical language. The reason for this is that the static character of classical logic makes it hard to use for describing programs having dynamic meanings. (At the MFCS '78 conference P. van Emde Boas gave an interesting review of the recent attempts for constructing non-classical logics of programs). Several talks in Salgótarján dealt with algorithmic logic, a non-classical logic worked out by Polish mathematicians (*L. Banaohowski, H. Basiowa, A. Salwicki* and others) in the late sixties.

In algorithmic logic programs are terms, and a new type of formula is introduced: if K is a program, and p is a formula, then Kp is also a formula. Kp is true, if p is true after the termination of K. A similar logic was constructed at MIT, the so called dynamic logic. As *A. Kreczmar* pointed out, the main difference between these two logics is that dynamic logic considers programs as relations, not as functions, to be able to cope with nondeterministic programs. Here $[K]p$ means that p is true after each execution of K, and $< K > p$ that there is an execution of K which makes p true. One can see that non-determinism gives rise to a kind of modality. The same trick can be done also in the frame of algorithmic logic, as *G. Mirkowska's* talk showed.

M. Grabowski showed that algorithmic logic can be expressed in infinitary or second order logic.

Beside the question of expressive power of the semantics describing language, the existence of a proper *calculus* is a pivotal question. Any kind of logic speaking about programs has to make us able to prove properties of programs. In the field of program verification one of the most important recent results is that there is no complete calculus to prove correctness of programs. This forces several computer scientists to weaken the notion of proof. The dynamic logic takes non recursive axiom system (all the formulas valid in the standard model of arithmetic), algorithmic logic has an infinite rule of inference to handle program loops, a version of the ω-rule. So the completeness of these logics can he stated, but the definability of the notion of proof is lost, that is it cannot be decided whether a piece of text is a proof or not. Moreover the notion of completeness is weakened: the set of theorems does not remain recursively enumerable.

H. Andréka, I. Németi, T. Gergely started along another way. (They and their coworkers will be referred to as AGN group). Their starting point is the first order classical logic. They studied why complete calculus for programs could be found and tried to characterize the class of provable programs. Their model theoretical research showed that the cause of incompleteness is the role of some kind of time scale in the semantics of programs – since their meaning is established by their executions. This role was made explicit: the language was extended by formulas referring to time. *T. Gergely's* talk summarized the effects of this approach. He pointed out that the usual way of analysis assumed that the execution of programs was according to a time scale isomorfic with natural numbers. So every assertions about programs involves the standard model of arithmetic, as he expressed it: "the metalanguage is the language of ω-logic". To avoid the incompleteness caused by it, non standard models of axiom systems of arithmetic have to be allowed to play the role of time scale, that is programs may terminate in "non-standard" time. This is somehow against oar intuition about programs, and forces us to alter some of our ideas. As an example: only a generalized theory of recursive functions can be applied. (See *L. Ury's* talk). However this theory makes properties of programs treatable in the frame of mathematical logic. *H. Andréka* showed, also using non-standard time scale, that the method of inductive assertions is complete. Moreover, she proved that a weaker condition does not ensure completeness. After the axiomatization of the method of inductive assertions (done by Z. Manna) this is the most important result in the theory of the method.

However, in spite of the results, the novelty of the AGN group's approach makes it hard to accept. *A. Salwicki* argued against the necessity of speaking about any kind of time scale. Their discussions made it dear that two antagonistic views met at the colloquium. Algorithmic logic built the notions of computing techniques into mathematical logic, not only to study programs, but also to break Into mathematics. According to *Salwicki*, they would like to see an algorithmic centred, constructive mathematics:

"mathematics and computer science are different facades of the same building". The AGN group views this building from the other side. They do not want to make mathematics more algorithmic, bat smuggle the tools of abstract mathematics into the analysis of programs. Maybe, it seems that algorithmic logic is more practical for those in programming practice.

However if we look at practical consequences of the two theories, we find that the use of the ω-rule algorithmic logic's proof procedures demands high quality of mathematical knowledge and a lot of invention. On the other hand, the ASN group uses sophisticated mathematical notions to give a correct mathematical theory for finitary, simply usable tools of programming.

The aim of research activities in computer science is not only to make it clear what programs and programming are, but also to give a theoretical basis and methods for a future programming technology. Only *J. Aszalós* talked directly about programming technology. He did not give a new theory or formalism, but showed how a system programmer, who is familiar with mathematical logic, could use it in his practical work.

In the last few years the greatest success of logic in programming has been the PROLOG language. *PROLOG* is a programming language based on logic. It is the first practical application of the principle "logic as a programming language". The underlying idea is that a certain set of axioms can be considered as a program," and a theorem to prove as the actual task for the program. If we have an automatic theorem prover effective enough, it can play the role of the interpreter.

And in the case of Horn formulas the resolution principle proved itself effective enough. PROLOG is widely applied for artificial inteligence problems. In Hungary it has stepped out of the AI laboratories, and it is used in everyday computations. The talks given by *K. Bruynoogh, L. Pereia*, and *P. Szeredi* showed that they do their best to improve PROLOG. *F. Darvas* and his coauthors' talk informed us about applications in the drug industry.

Realising that the colloquium showed a certain gap between theory and practice, the organising committee set up "logic and practice" as the subject for the panel discussion. However it soon turned into a discussion on the computing of the future. First we thought that there was a tacit consent about the necessity of making computer systems mare intelligent, but several of the participants did not think this possible because of complexity bounds. They did not realise that not the problems themselves, but the methods for their solution were characterised by these bounds. Moreover, the members of the algorithmic logic school did not want to work on developing more intelligent systems. For them *assignment* and *iteration*, as the underlying ideas of the present programming, meant the top of the culture of programming. However computer scientists working on PROLOG pointed out how *pattern matching* and *backtracking* opened a wide horizon for new possibilities.

We reviewed the colloquium from a special point of view, therefore we omitted several interesting talks. Such were *O. Stepankova's* on the subject of automatic problem solving, *L. Farinas del Cerro's* about mechanization in non classical logic and several others. The papers will be published by North Holland in the series "*Colloquia Mathematica Societatis János Bolyai*".

Authors:

Edit Sánta-Tóth
SzKI, Budapest, Adadémia u. 17. 1054.
HUNGARY

Miklós Szöts
SzÁMKI, Budapest, Pf. 227. 1536.
HUNGARY