Author


Professor Jean-Baptiste Lagrange
Université Paris-Diderot & University of Reims, Laboratoire de Didactique André Revuz, Paris, France The final publication is available at Springer http://rd.springer.com/referenceworkentry/10.1007/978-94-007-4978-8_7

Editor


Professor Stephen Lerman
London South Bank University, Department of Education, London, UK

Algorithmics

Encyclopedia of Mathematics Education · Article ID: 313186 · Chapter ID: 7

Key-words

Algorithmics, Algorithms, Programming, Computer Sciences, Programming Language

Definition

"Algorithmics" can be defined as the design and analysis of algorithms (Knuth 2000). As a mathematical domain, algorithmics is not principally concerned by human execution of algorithms for instance for arithmetic computation (see 2010/index/chapterdbid/313187 for a discussion) but rather by a reflection on how algorithms are built and how they perform. Algorithms exist and have been studied since the beginning of Mathematics. However, the emergence of algorithmics as a mathematical domain is contemporary to digital computers, the work on computability by Church (1936), Turing (1937) and other mathematicians being often considered as seminal. Computer science, also emerging at the same time, is concerned with methods and techniques for machine implementation whereas algorithmics focuses on the properties of algorithms.

Typical questions addressed by Algorithmics are the effectiveness of an algorithm (whether or not it returns the expected result after a finite number of steps), the efficiency (or complexity) of an algorithm (an order of the number of steps for a given set of data) and the equivalence of algorithms (for example iterative and recursive equivalent forms). Djiskra (1976, p 7) notes that "as long as an algorithm is only given informally, it is not a proper object for a formal treatment" and therefore that "some suitable formal notation" is needed "to study algorithms as mathematical objects". This formal notation for algorithms or "language" is a vehicle for abstraction rather than for execution on a computer.

Algorithms in mathematics education research

Research in Mathematics education and computers most often concentrates on the use of technological environments as pedagogical aids. Authors like Papert and Harel (1991), Dubinsky (1999) or Wilensky and Resnick (1999) proposed computer programming as an important field of activity to approach mathematical notions and understanding. This strand of research does not consider the design and analysis of algorithms as a goal in itself. The hypothesis is that building algorithms operating on mathematical objects and implementing these in a dedicated programming language (LOGO or ISTL) is able to promote a "constructive" approach to scientific concepts. The language's features (recursivity, functions...) are chosen in order to support this approach. Students' access to a formal algorithmic language is generally not an issue because the tasks proposed for students generally imply short programs with a simple structure.

In a few countries and regions, curricula for Algorithmics have been implemented and, in parallel, research studies have been conducted. For instance, at the end of the years 1980, a curriculum has been written and tested for 7th and 8th grade students in a region of Germany (Cohors-Fresenborg, 1993). Concepts of Algorithmics were taught by making students solve calculation problems using of a concrete "registermachine".

These research studies are few and do not really tackle questions at the core of algorithmics like effectiveness and complexity, reflecting the fact that, at school levels investigated by research studies, students' consideration of algorithmics is still limited by the difficult access to a symbolic language.

Students' understanding of algorithmic structures and languages

In France, programming algorithms has been proposed as a task for secondary students in various curricula. Because the time devoted for these tasks was short, students' understanding of algorithmic structures and languages appeared to be the real challenge, algorithmics in the sense of Knuth (2010) being inaccessible to beginners without this prerequisite. Didactic research studies were developed focusing on this understanding.

Samurcay (1985) was interested by 10th grade students' cognitive problems relatively to variables in iteration. The method was to ask students to complete iterative programs in which instructions were missing. Missing instructions were of three types: the initialization of the iterative variable, an assignment of the iterative variable in the loop body, and the condition for exiting the loop. Important misunderstandings of the semantics of variables were identified. For instance, regarding the initialisation, some students think that the initial value has necessarily to be entered by a reading instruction; others systematically initialise variables to zero. They are clearly influenced both by preconception of how a computer works, and by previous examples of algorithms that did not challenge these preconceptions. The author concludes that more research studies are essential in order to understand how students conceptualize the notions associated to iteration, and to design adequate didactical situations.

Samurcay, Rouchier (1990) studied students' understanding of recursive procedures distinguishing between two aspects: self-reference (relational aspect) and nesting (procedural aspect). They designed teaching sessions with the aim to help pupils to construct a relational model of recursion, challenging students' already existing procedural model. After sessions of introducing the students to the LOGO graphic language without recursion, they designed ten lessons: first introducing the students to graphic recursive procedures, making them distinguish between initial, central, and final recursion and then
helping them to generalise recursive structures by transferring recursive procedure to numerical objects for tasks of generating sequences. Observing students, they conclude that introducing recursion is a non obvious "detour" from already existing procedural model of iteration, and a promising field for research.

Lagrange (1995) considered the way 10th and 11th grade students understand representations of basic objects (strings, Booleans) in a programming language. Analysing students' errors in tasks involving simple algorithmic treatments on these objects, he found that misunderstandings result from assimilation to "ordinary" objects and treatments. For instance, when programming the extraction of a substring inside a string, students often forgot to assign the result to a variable; the reason is that they were not conscious of the functional nature of the substring instruction, being influenced by the "ordinary" action oriented language. Another example is that students generally did not consider the assignment to a Boolean value, not understanding that in an algorithmic language, "conditions" are computable entities. Similar difficulties found in this study were analysed in relationship with analogous obstacles in accessing the algebraic symbolism at middle school level. Programming simple algorithms involving these non numerical objects seemed promising for overcoming such obstacles.

Nguyen (2005) questioned the introduction of elements of algorithmics and programming in the secondary mathematical teaching, showing that, on one hand there is a fundamental solidarity between mathematics and computer science based on the history and the current practice of these two disciplines, and that on the other hand, the ecology of algorithmics and programming in secondary teaching is not obvious. Focusing on the teaching/learning of loop and of computer variable notions in France and in Vietnam, he proposed an experimental teaching unit in order that 10th grade students learn the iterative structure. He chose to make students build suitable representations of this structure by solving tasks of tabulating values of polynomial using a dedicated calculator, emulated on the computer, and based on the model of calculator existing in the secondary teaching of the two countries with the additional capacity to record the history of the keys pressed.

The experimental teaching was designed as a genesis of the machine of Von Neumann: the students had to conceive new capabilities for the calculator especially erasable memories and controlled repetition in order to perform iterative calculations, and programming through the writing of the successive messages (programs) to machines endowed with different characteristics. This allowed for the emergence of the notion of iterative variables and treatments. In the framework of the Theory of Didactical Situations, a milieu and a fundamental situation are then offered for the construction of the iterative structure.

Algorithmics and programming competencies

In parallel to mathematics education research, studies have been carried out in the field of psychology of programming. Most studies in the field address professional programming and discuss opportunities and constraints of programming languages, and design strategies for experts (for an example, see Petre & Blackwell 1997). Some studies focused on programming problem solving by beginners with tasks very close to students' activity in early algorithmics courses. For instance Rogalski and Samurçay (1990) focused on the acquisition of programming knowledge "as testified by students' ability to solve programming problem", that is to say, to pass from "real" world objects and situations to an effective program implementation. Rogalski and Samurçay (1990) insist on "the variety of cognitive activities and mental representations related to program design, program understanding, modifying, debugging (and documenting)". They stress the necessity for beginners of adequate mental models of data representation and processing.

These models include static schemas and plans. Schemas are defined as sets of organised knowledge used in data processing that help to achieve small scale goals. Plans are organized sets of dynamic procedures related to the schemas. For instance, when programming the sum of numbers in a list of arbitrary length, schemas are related to different sub tasks like entering the list and computing iteratively partial sums, and the plans help to define a strategy, separating the two sub tasks, or merging these in a single iteration. More generally, research in the field of psychology of programming by beginners usefully complements math education research because it introduces theoretical models of human thinking to give account of competencies required to build or understand programs or algorithms.

Perspectives

In spite of nearly 30 years of existence, mathematics education research in algorithmics remains in its infancy. It is conditioned by political decisions to include algorithms in the mathematics curriculum. Finding ways to help students access an algorithmic language together with adequate mental models of data representation and processing appears to be a condition in order that they could tackle central questions like complexity or proof of algorithms. This is consistent with Djiskra's (ibid.) epistemological view that a suitable formal notation is needed to study algorithms as mathematical objects. It is also a stimulating challenge that the above mentioned research studies just started to take up.

Cross-References

Algorithms

References

Cohors-Fresenborg, E. (1993) Registermaschine as a Mental Model for Understanding Computer Programming. In E. Lemut, B. du Boulay, G. Dettori (Eds.), *Cognitive Models and Intelligent Environments for Learning Programming*, 235-248. Berlin: Springer.

Church, A. (1936) An unsolvable problem of elementary number theory; American Journal of Mathematics 58, 345-363.

Djiskra, E., D. (1979) *A discipline of programming.* Prentice-Hall, Englewood Cliff.

Dubinski, E. (1999) One theoretical perspective in undergraduate mathematics education research, In O. Zaslavsky (Ed.), *Proceedings of the 23rd Conference of PME*, Haifa, Israel, 4, 65-73.

Lagrange J.B. (1995) Bridging a GAP from Computer Science to Algebra. In *Technology in Mathematics Teaching* L. Burton & B. Jaworski (eds.) Chartwell-Bratt 1995

Nguyen C. T. (2005), *Étude didactique de l'introduction d'éléments d'algorithmique et de programmation dans l'enseignement mathématique secondaire à l'aide de la calculatrice*, Thèse de l'université Joseph Fourier, Grenoble

Papert S., Harel, I. (1991) *Constructionism.* Ablex Publishing Corporation

Petre, M., & Blackwell, A. F. (1997). A glimpse of expert programmers' mental imagery. In *Papers presented at the seventh workshop on Empirical studies of programmers* (pp. 109-123). ACM. Retrived from https://www.cs.duke.edu/courses/fall00/cps189s/readings/petre-expert.pdf

Rogalski, J., & Samurçay, R. (1990). Acquisition of programming knowledge and skills. In J.-M. Hoc, T. G. R. Green, R. Samurçay, & D. Gilmore (Eds.), *Psychology of Programming* (pp. 157-174). Londres : Academic Press.

Samurcay, R., (1985) « Signification et fonctionnement du concept de variable informatique chez des élèves débutants », *Educational Studies in Mathematics* n°16 pp 143-161

Samurcay, R., Rouchier, A., (1990) « Apprentissage de l'écriture et de l'interprétation des procédures récursives », *Recherches en didactique des Mathématiques*, Vol 10 2.3. pp 287-327

Turing, A. M. (1937) *On Computable Numbers, with an Application to the Entscheidungsproblem*, Proc. London Math. Soc., 2^{e} série, 42, 230-265

Wilensky, U., Resnick, M. (1999) Thinking in levels: A dynamic systems approach to making sense of the world. *Journal of Science Education and Technology*.8(1) 3-19