Annals of Mathematics and Artificial Intelligence (2023) 91:563–565
https://doi.org/10.1007/s10472-023-09846-1

Preface

Adel Bouhoula[1] · Bruno Buchberger[2] · Tetsuo Ida[3] · Temur Kutsia[2]

Accepted: 22 March 2023 / Published online: 2 May 2023

This special issue of Annals of Mathematics and Artificial Intelligence is devoted to theoretical and practical aspects of symbolic computation in software science, combined with recent artificial intelligence techniques. Symbolic Computation is the science of computing with symbolic objects (terms, formulae, programs, representations of algebraic objects, etc.). Powerful algorithms have been developed during the past decades for the major subareas of symbolic computation: computer algebra and computational logic. These algorithms and methods are successfully applied in various fields, including software science. Meanwhile, artificial intelligence methods and machine learning algorithms are widely used nowadays in various domains and, in particular, combined with symbolic computation. Several approaches mix artificial intelligence and symbolic methods and tools deployed over large corpora to create what is known as cognitive systems.

Software science, broadly understood, covers a wide range of topics, among others: construction, analysis, transformation, and verification of software; programming models; formal methods for software development; creation and processing knowledge libraries; etc. This special issue was open to papers about software science-relevant techniques and methods originating from logic, algebra, and artificial intelligence. We also welcomed submissions related to knowledge formalization, large-scale computer understanding of mathematics and science, cognitive computing, etc. Three papers, reflecting various aspects of this understanding of the relationship between symbolic computation, software science, and AI, constitute this issue.

In the first article, *Bruno Buchberger* presents his view on the interaction between automated programming, symbolic computation, and machine learning. Automated programming concerns procedures that (semi-)automate various aspects of the programming

✉ Temur Kutsia
kutsia@risc.jku.at

Adel Bouhoula
a.bouhoula@agu.edu.bh

Bruno Buchberger
Bruno.Buchberger@risc.jku.at

Tetsuo Ida
ida@cs.tsukuba.ac.jp

1 Arabian Gulf University, Manama, Bahrain

2 Research Institute for Symbolic Computation, Johannes Kepler University, Linz, Austria

3 University of Tsukuba, Tsukuba, Japan

process (“meta-programming”). Buchberger considers automated programming as the “most exciting and relevant technological endeavor today” that will have “an enormous impact on the global job market in the software industry”, and looks at symbolic computation and machine learning as two fundamentally different approaches to automated programming. In his analysis, both approaches are part of algorithmic mathematics and they, as mathematical methods, require quite some human intelligence while they are being invented or developed. However, applying the already invented methods can hardly be considered to be a task that requires “intelligence” (since it is just an execution of algorithms) and, hence, calling such an application “machine intelligence” or “artificial intelligence” is misleading, argues the author. Besides the general characterization of the approaches, Buchberger also reports about his experience in automated programming by symbolic techniques (based on his own “lazy thinking” algorithm synthesis method) and by a machine learning approach (synthesizing programs from natural language specifications using ChatGPT). He concludes the paper with the message that “the next big step forward could and should be to combine the machine learning approach to generating (maybe not completely perfect) programs from natural language specifications and the symbolic computation approach to handling various intermediate steps in the programming process”, describing also a possible way how such a combination can be achieved.

The second article, authored by *Johannes Blümlein*, *Marco Saragnese*, and *Carsten Schneider*, describes new symbolic tools to gain a large-scale computer understanding of processes in quantum chromodynamics. For the precision calculations in this area, huge expressions of several GB in size have to be dealt with. The goal is to obtain compact representations of them that are suitable for further processing and provide further insight within the arising calculations. More precisely, highly complicated divergent multi-loop Feynman integrals in the given expressions have to be calculated analytically in order to represent the expressions more compactly, in terms of special functions and constants. The input is given in the form of very large sets of (coupled partial) linear differential equations, and the authors describe a general toolbox for solving them symbolically. They also propose a systematic classification of partial differential equations for scalar or master integrals (of one or more scales) with respect to known solutions in the hypergeometric classes. These developments can be considered as the first steps towards the computerization of knowledge about special function-based result representations in quantum chromodynamics.

In the third article, *Sorin Stratulat* proposes a solution to the problem of validation of cyclic pre-proofs produced by software: cyclic induction reasoner. Cyclic induction is a reasoning technique used to terminate the proof attempt of certain subgoals if they have been already generated earlier during the proving process. For first-order logic with inductive definitions and equality, CLKID$^{\omega}$ is the inference system that aims at building (cyclic) proofs using this technique. Some cyclic derivation trees constructed by CLKID$^{\omega}$ in the proving process might not be sound. Therefore, such trees in general are called pre-proofs rather than proofs. CYCLIST is a theorem prover that implements CLKID$^{\omega}$. Pre-proofs produced by CYCLIST require human validation, which might be quite tedious. To address this problem, Stratulat proposes an approach that allows certifying cyclic pre-proofs mechanically. It is used to certify pre-proofs produced by E- CYCLIST (a prover that extends CYCLIST) using the Coq proof assistant.

We thank the authors who submitted their papers to this issue and the reviewers who evaluated the submissions thoroughly and helped the authors to improve their contributions.

Special thanks go to Martin Golumbic, the editor-in-chief of Annals of Mathematics and Artificial Intelligence, who was very helpful throughout the editorial process.

Data Availibility This manuscript has no associated data

Declaration

Conflicts of interest The authors declare that they have no conflict of interest