Journal of Automated Reasoning (2019) 63:809–811
https://doi.org/10.1007/s10817-019-09529-9

Preface to the Special Issue on Linearity

Iliano Cervesato[1] · Maribel Fernández[2]

Received: 12 July 2019 / Accepted: 15 July 2019 / Published online: 27 July 2019

Since the birth of linear logic, there has been a stream of research where linearity is a key issue, covering both theoretical topics and applications to several areas of Computer Science, such as work on proof technology, complexity classes and more recently quantum computation, program analysis, expressive operational semantics, linear programming languages, and techniques for program transformation, update analysis and efficient implementation.

The wealth of recent results and the opening of new, exciting, research directions in the area led to the introduction of a series of international workshops. The LINEARITY workshops bring together researchers who are developing theory and applications of linear calculi. Topics of interest include: sub-linear logics, linear term calculi, linear type systems, linear proof theory, linear programming languages, applications to concurrency, interaction-based systems, verification of linear systems, quantum models of computation, and biological and chemical models of computation.

This special issue is devoted to papers describing recent advances in this area. It came out of a call for papers issued after the fourth International Workshop on Linearity, with the aim of including a selection of extended versions of workshop papers in addition to opening it up to other authors. Six articles were selected for this special issue, covering the following topics:

1 Implicit Computational Complexity

Linear type systems are a useful tool in the analysis of the complexity of higher-order programs. The article by Baillot, Dal Lago and Barthe, *Implicit Computational Complexity of Subrecursive Definitions and Applications to Cryptographic Proofs*, defines a linear dependent type and effect system for a call-by-value variant of Godel's System T with references, which can estimate the complexity of programs as a function of the size of their inputs. The authors describe a sound and complete type inference procedure that over-approximates the

✉ Iliano Cervesato
iliano@cmu.edu

✉ Maribel Fernández
maribel.fernandez@kcl.ac.uk

1 Carnegie Mellon University, Pittsburgh, USA

2 King's College London, London, UK

complexity of executing the programs on a variant of the CEK abstract machine. The power of this result is illustrated on several examples of cryptographic proofs.

2 Model Checking

Model checking and theorem proving are two classic approaches to certifying the correctness of programs and systems. Although both are rooted in logic, they largely occupy separate worlds, the former stemming from model theory, the latter from proof theory. The paper by Heath and Miller, *A Proof Theory for Model Checking*, takes a step towards bridging these two worlds by exploring proof-theoretic support for model checking techniques. Specifically, they leverage the interplay between additive and multiplicative connectives in focused linear logic to explain notions such as reachability, non-reachability, as well as tabled deduction, bisimulation, and winning strategies.

3 Numeral Systems

Mackie's article, *Linear Numeral Systems*, provides an answer to a long-standing question, namely the possibility of defining efficient numeral systems in the linear lambda calculus, where terms cannot be copied or erased. The paper addresses three problems: how to represent numbers, how to define constant time arithmetic operations for successor, addition and predecessor, and how to define subtraction in an efficient way. The latter is well-known to be difficult in numeral systems.

4 Proof Diagrams

Proof nets, a graphical syntax for linear logic proofs, are a popular tool to represent and analyse proofs: their graphical nature avoids the need to explicitly define some proof equivalences that are needed when using textual representations. Acclavio's article, *Proof Diagrams for Multiplicative Linear Logic: Syntax and Semantics*, proposes an alternative 2-dimensional syntax for multiplicative linear logic derivations using string diagrams. In this approach, deciding whether a term corresponds to a correct proof derivation is efficient: the check can be done in linear time, thanks to the fact that string diagrams permit to include control strings to encode the correct application of inference rules. Moreover, the paper shows how to define a denotational semantics for multiplicative linear logic proofs with units, using equivalence classes of proof diagrams.

5 Quantum Programming Languages

In the approaches to the design of quantum programming languages that exist in the literature, linear logic plays a major role. Many of the languages are based on a linear algebraic lambda-calculus, or use linear types to model features of the quantum computation. Two articles in this special issue study quantum languages. In the article by Paolini, Piccolo and Zorzi, *QPCF: higher order languages and quantum circuits*, a quantum programming language is built on top of PCF, by adding quantum circuits and a quantum co-processor. A dependent type system is used to deal with quantum circuits in a way that is completely different from the standard approaches, based on linear types. Using a big-step operational semantics, the authors show that despite the extension with quantum features, the language satisfies standard evaluation properties and is sufficiently expressive.

In the article by Mahmoud and Felty, *Formalization of Metatheory of the Quipper Quantum Programming Language in a Linear Logic*, a linear logical framework is defined within the Hybrid system in order to analyse the type system of a quantum lambda calculus called Proto-Quipper, which contains the core of Quipper. To facilitate reasoning on Quipper's linear type system, the authors introduce a linear specification logic and formalise the typing and evaluation rules of Proto-Quipper in this logic. The article shows the type soundness of Proto-Quipper and discusses the adequacy of the encodings.