ELSEVIER

Theoretical Computer Science 199 (1998) 1–3

Theoretical
Computer Science

Editorial

The papers in this volume are revised and extended versions of communications presented at the First International AMAST Workshop on Algebraic Methods in Language Processing (AMiLP '95), held at the University of Twente, Enschede, the Netherlands, 6–8 December 1995. The workshop was organized in the framework provided by the Algebraic Methodology and Software Technology (AMAST) movement. In this framework large international conferences and specialized workshops are held. The AMiLP workshop considered algebraic methods in formal languages, programming languages and natural languages. Responsible for the scientific program of the workshop were Anton Nijholt (Enschede), Maurice Nivat (Paris), Teodor Rus (Iowa City), and Giuseppe Scollo (Enschede).

The presentations in the workshop were by invitation only. Not all presentations are reflected by papers in this special issue. Tutorial presentations were given by, among others, Michael Moortgat on categorial grammars, Grzegorz Rozenberg on theories of text and Robert Berwick on "minimalist" syntax. Some of them were based on material that in the mean time has been published elsewhere.

The aim of the workshop was to bring together researchers on formal language theory, programming language theory and natural language description theory, who have a common interest in the use of algebraic methods to describe syntactic, semantic and pragmatic properties of language. The workshop demonstrated that there is interesting use of algebraic methods in natural language description (e.g., categorial grammar), programming language processing (e.g., compiler construction and development of programming language environments) and (obviously) in formal language theory.

Language processing is one of the oldest and still a very lively topic in software technology. Formal language theory finds its origin in the development of grammar formalisms for natural and programming languages. Formal description of the syntax and semantics of programming languages has required intensive research on the mathematical foundations of the models used. Formal descriptions of concurrent systems have led to research on formal specification languages and a need to have means for describing syntactic, semantic and pragmatic features of these languages.

Traditionally, automatic generation of tools (e.g., compilers) and of programming environments has been based on algebraic concepts and methods. Recently, in software technology there is growing interest in the role of natural language. The first reason is the role natural language plays in the traject of requirements engineering to formal problem specification. The second reason is that natural language is the most natural way people express themselves and therefore, in human-computer communication natural language cannot be dismissed. The same holds for computer-

PII S0304-3975(97)00265-X

mediated human-to-human communication. The third reason is that due to progress in the field of speech and (natural) language engineering software technology has to be more and more concerned about applications in which natural language transformation is the main problem. Applications range from all kinds of document processing, optical character recognition, machine translation, natural language interfaces, information retrieval and text processing.

Formal description of natural language has led to the use of proof-theoretic systems and to the investigation of their algebraic foundations. Categorial grammar with its algebraic semantics and the application of Boolean algebras can serve as an example. In order to represent meaning, a wide variety of algebraic models has been introduced. Recently, in describing natural language (utterances and dialogues) there is interest in algebras (process algebra, module algebra, etc.) that have been developed in the field of the foundations of software engineering. We are thus facing a rich cross-fertilization between several disciplines, where the use of algebraic methods appears to be the main catalyst. We hope the papers in this volume will further promote this process.

Papers in this volume

The paper *Algebraic Structures in Categorial Grammars* by Wojciech Buszkowski gives a tutorial introduction to the author's research on different algebraic structures. These prove to be natural frameworks for characterizing languages of different kinds of categorial grammars, ranging from basic categorial grammars to the language hierarchies of Lambek categorial grammars. In these frameworks algorithms are provided for solving equivalence problems and related questions, and for developing learning procedures.

In *Algebraic Translations, Correctness and Algebraic Compiler Construction*, Theo Janssen argues for the versatility of algebraic methods as it appears in several fields where a translation problem occurs. On the natural language side he summarizes the algebraic basis of the Rosetta machine translation project. More generally, he focuses on the notion of correctness of translation by comparing different proposals from the literature and assessing their relative adequacy by means of examples.

Eelco Visser's paper on *Polymorphic Syntax Definition* investigates the two-level extension of the well known correspondence between context-free grammars and first-order signatures. After giving examples of the usage of two-level grammars for polymorphic syntax definition the author identifies a subclass of two-level grammars with a decidable parsing problem, provides an algorithm for this, and shows its correctness.

Parsing Schemata and Correctness of Parsing Algorithms by Klaas Sikkel introduces his parsing schemata framework, a high-level formal description of parsers which can be used to formalize relations between different parsers by relating their underlying schemata. The framework allows variants, extensions and optimizations to be exchanged across algorithms. Moreover, it can be used as an intermediate level of abstraction for deriving the formal correctness of a parser.

A tutorial introduction to an algebraic methodology of compiler construction is provided by Teodor Rus in his paper *Algebraic Processing of Programming Languages*. This methodology, which arises from experience with "real world" programming languages, is nevertheless based on a formal concept of programming language that is presented here. Because of the compositional character of the algebraic approach, this methodology succeeds in separating independent components of programming languages. This enables independent development of corresponding tools and their smooth integration in resulting compilers.

Algebraic power series are the main ingredient in Frédéric Tendeau's paper *Computing Abstract Decorations of Parse Forests Using Dynamic Programming and Algebraic Power Series*. The formalism is made use of, to show how to apply dynamic programming techniques to compute decorations in an abstract semiring. Useful instances of this structure are the boolean semiring (for recognition), a parse forest semiring (for parsing) and a semiring of probabilities (for stochastic parsing).

The paper *Information Flow in Tabular Interpretations for Generalized Push-Down Automata* by Eric Villemonte de la Clergerie and François Barthélemy introduces and analyzes a generalization of push-down automata that they claim to be a general framework for deriving tabular algorithms for a large class of stack based computations. This operational framework can be made use of to implement syntactic, logic or constraint based formalisms, and enables the transfer of techniques between them.

Phrase Parsers from Multi-Axiom Grammars by Teodor Rus and James S. Jones argues for the use of multi-axiom grammars whereby the distribution of the parsing task among a hierarchy of language layers naturally arises. The layers are determined by the algebraic properties of these grammars. Algorithms are given for this stratification and for the construction of parsers for each layer that interwork in a pipeline parallel fashion.

The final paper of this volume, *Algebraic Specification of Documents* by José Carlos Ramalho, José João Almeida and Pedro Henriques, presents a concrete problem domain, viz. that of document processing, where they propose an algebraic approach to define document types and to specify document manipulation. Their proposal is based on the use of their Camila system which is a constructive specification framework that allows one to build and run program prototypes.

Acknowledgement

Our words of thanks are addressed to the institutions and people who made AMiLP '95 and this special issue possible; in particular to the authors and the referees for their efforts towards meeting stringent quality requirements, and to Maurice Nivat, Editor-in-Chief of *Theoretical Computer Science*, for his support to bringing this special volume into existence.

A. Nijholt and G. Scollo
Guest Editors