ELSEVIER

Theoretical Computer Science 232 (2000) 1–4

Theoretical Computer Science

www.elsevier.com/locate/tcs

Preface

Algorithmic proof-search is a fundamental enabling technology throughout computing science. There is a long history of work in proof-search in a variety of systems of logic, including classical, intuitionistic, relevant, linear and modal systems, at the propositional, first- and higher-order levels. Such work has ranged from the most abstract to the most practical and has employed the full spectrum of logical techniques, from proof theory, model theory and recursion theory.

Recently, there has been a great deal of work on proof-search in type-theoretic languages. Such languages can be thought of as logical frameworks to represent proofs and to formalize connections between proofs and programs. Two workshops on *Proof-search in Type-theoretic Languages* (Nancy in 1994 and Rutgers University, NJ, in 1996) have already provided exchanges of ideas and experience in topics concerned with proof-search in type theory, logical frameworks and their underlying (e.g., classical, intuitionistic, linear) logics. Here again, the scope of languages studied and techniques employed has been wide, stretching to include algebraic and categorical methods. From the computational point of view, the type-theoretic component of logical languages, which may involve propositional, first-order, higher-order or polymorphic assignment régimes, introduces significant challenges for both theoreticians and implementors.

The topics of interest mentioned by the Call for Contributions to this Special Volume on *Proof-search in Type-theoretic Languages* included, but were not restricted to:

- Natural deduction, sequent calculi systems for type-theoretic languages;
- Tableaux, matrix or resolution methods for proof-search in type-theoretic languages;
- Semantic techniques in proof-search, search *versus* deduction as the basis of logic, consequences for model theory;
- Theorem proving and program development with type-theoretic languages: concepts, techniques, implementation and experimentation;
- Logic programming in type-theoretic languages as search-based computation; integration of model-theoretic semantics and imperative aspects of logic programming;
- Operational semantics and proof theory of search-based computation; denotational semantics and model theory of search-based computation;
- Complexity of search problems in type-theoretic languages; comparisons with non-type-theoretic systems.

The papers selected for this Special Volume address many of these topics, with foci on problems and topics, such as:

- How to model cut-free arithmetic for interfaces with existing proof-engines;
- The design of algorithms for provability or unprovability in many implicational logics;

PII: S0304-3975(99)00168-1

- A constructive approach, leading to proved-correct procedures, for various search problems;
- The efficient management of linear contexts in linear logic programming;
- For non-trivial applications, the adaptation of a well-known procedure to increase the automation of proof-search in the Calculus of Constructions;
- The use of proof-nets as an alternative tool for automated deduction in linear logic;
- Reasoning about specifications of computation with an intuitionistic logic that includes definitions and induction;
- Characterization of cases in which classical provability and intuitionistic provability coincide and their relationships with uniform provability;
- Proof-search in intuitionistic logic viewed as a pertubation on proof-search in classical logic via a type-theoretic, and hence semantic, calculus.

Although the *proofs-as-programs* paradigm is naturally mentioned, or indeed used, in many works on type theory, papers dealing with a non-trivial application of this paradigm are all too rare. The paper *Search Algorithms in Type Theory* applies this technique to the derivation of a sophisticated search algorithm. From the analysis of a large family of search problems and their specifications, a constructive proof is presented which has as its computational content a family of correct search procedures for the family of search problems. By an extension based on the classical typing of control, this proof can incorporate a sophisticated backtracking technique ("conflict-directed backjumping"). It can also be formalized in the Nuprl system.

There is a difference between specifying a computation and reasoning about a computation, via the proof-search paradigm. A possible approach for such reasoning consists in introducing new inference rules both for induction on the natural numbers and for treating logical specifications as definitions in the sequent calculus of intuitionistic logic. The paper *Cut-elimination for a Logic for Definitions and Induction* presents such a new sequent calculus by focusing on the cut-elimination theorem that is important from different points of view: consistency of the logic; automation of proof-search. The key features of the calculus (induction, definitions) and their interactions lead to a new cut-elimination proof that is technically interesting and significantly extends previous results of this kind.

In some type-theoretic languages which include higher-order quantifiers, arithmetic can be reduced to logic (considering, for instance, the set of naturals as the intersection of all sets containing 0 and closed under successor). Then several known strategies for higher-order proof-search can be traced to strategies for arithmetic. In this context, the paper *Extended Normal Form Theorems for Logical Proofs for Axioms* proposes a strategy for proof-search in cut-free logic which allows an exact modelling of cut-free arithmetic. Moreover, exact proof-theoretic analysis reveals connections with omega-consistency and reflection.

The design of programming languages and theorem provers based on linear logic opens new implementation challenges not present in more traditional languages. For their use in non-trivial applications, the problem of efficiently managing the linear context is of crucial importance. The paper *Efficient Resource Management for Linear*

Logic Proof Search studies this problem within the Lolli system with results having application to other systems based on linear type theory. It presents resource-management systems designed to eliminate the non-determinism in the distribution of linear formulae and, consequently, improve the efficiency of implemented systems through an improved proof-search process.

In the context of the proof-as-programs paradigm, more precisely of the encoding of proofs into λ-terms, finding proofs in fragments of intuitionistic logic corresponds to finding inhabitants of types. Previous work shows that for many implicational logics one can specify the set of all the λ-terms that represent proofs in such a logic. The paper *Proof Finding Algorithms for Implicational Logics* gives, for most of these logics, algorithms which return, for a given formula, a proof (if possible within a fixed number of steps) or otherwise a guarantee of unprovability. These new algorithms have been implemented in the *Oostdijk* system.

Another approach for proof-search improvements could be to consider a known procedure dedicated to a given logic (for instance classical logic) and without direct connection with type theory, to provide increased automation in proof-search in some logical frameworks? For instance, higher-order logic and higher-order type theories serve as the logical foundation of various theorem provers and little work has been done on providing the means for adapting proof-search method designed for one of them within the other.

The paper *The Calculus of Constructions as a Framework for Proof Search with Set Variable Instantiation* presents how a particular proof-search procedure (finding substitution instances for set variables) designed for higher-order logic can be used to improve the automation of proof-search in the Calculus of Constructions (CC). The initial techniques are incorporated and extended in a reformulation of an initial search procedure for CC and the new notion of "search contexts" for CC leads to a new proof-search procedure that can be adapted to useful "sublanguages", such as $\lambda\Pi$ (the language part of the LF logical framework) and higher-order hereditary Harrop (hohH) formulae.

Some logical frameworks and sequent calculi are based on fragments of non-classical logics, such as intuitionistic logic or linear logic, and result from compromises between the expressivity of the logic and the ability to automate proof-search efficiently. A way to improve proof-search in such logics (and their corresponding frameworks) and to have a better knowledge of their proof-theoretical properties could be to consider non-classical search (in intuitionistic or linear logics) as a perturbation on classical search. One possible approach consists in considering well-known provability characterizations and their connected efficient proof methods, such as the connection (or matrix) method for classical logic, and trying to define naturally similar methods for the given logic. In the paper *Connection Methods in Linear Logic and Proof Nets Construction*, some characterizations of provability based on connections are given for fragments of linear logic. Moreover, a connection proof-search method is designed from the automated construction of proof nets. The notion of proof net, the counterpart in linear logic of natural deduction, provides an interesting alternative approach to automated deduction

in linear logic. Applications in linear logical frameworks or theorem provers are a possibility.

In the analysis of the relationships between intuitionistic provability and classical provability, different approaches are possible. The paper *Correspondences between Classical, Intuitionistic and Uniform Provability* starts from an analysis of the inference rules used and provides a characterization of the cases in which classical provability entails intuitionistic provability. Moreover, these derivability notions are related with uniform provability, restriction of the intuitionistic one which is the basis of so-called abstract logic programming languages and of type-theoretic languages. The sense in which intuitionistic search can be viewed as a perturbation on classical search is illustrated in the paper *On the intuitionistic force of classical search* by taking a type-theoretic view of this approach. A system of realizers (proof-objects) for sequents in classical logic (the types) is developed by extending $\lambda\mu$-calculus with the aim of reflecting the properties of classical proof-search in the system of realizers. An application to a proof procedure, based on the extension of the notion of uniform proof to a multi-conclusioned sequent calculus, is presented.

In fact, the topics of these selected papers span, from various points of view and in different contexts, a wide range of the problems and issues related to the proof-search in type-theoretic languages. As Guest Editors, we should like to express our thanks to the authors for their high-quality contributions to this Special Issue and to the referees who have performed an important scientific task, essential for the selection of, and improvements to, the final papers. We are very grateful to Professor Maurice Nivat, Editor-in-Chief, who has given us the opportunity to realize this Special Issue on the topic *Proof-search in Type-theoretic Languages*. This topic is, we believe, at the frontiers of many different domains, considered here from the proof-theoretic point of view.

This volume presents some recent results which the potential to be applied throughout theoretical and applied computer science. It also raises some interesting issues. Perhaps foremost amongst these is the question of the rôle of semantic techniques in the theory of proof-search, particularly in type-theoretic settings. This is one of the main topics raised in our introductory article, “Proof-search in Type-theoretic Languages: an introduction”.

October, 1998

D. Galmiche
LORIA – Université Henri Poincaré, Campus Scientifique, B.P. 239, 54506 Vandoeuvre-les-Nancy Cedex, France

D.J. Pym
Queen Mary & Westfield College, Department of Computer Science, Mile End Road, London, E1 4NS, UK