Fundamenta Informaticae 24 (1995) 1–2
IOS Press

EDITOR'S PREFACE

David A. PLAISTED
Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, NC 27599-3175, U.S.A., e-mail: plaisted@cs.unc.edu and (Sabbatical), MPI fuer Informatik, Im Stadtwald, D-66123 Saarbruecken, Germany

The field of term-rewriting systems is now well-established, with a conference devoted to it every two years as well as significant contributions to other conferences such as CADE and LICS. What is surprising about term-rewriting systems is their structural simplicity and the depth of the resulting theory. A term-rewriting system is a set of equations, oriented from left to right. These are used to replace instances of their left-hand sides by corresponding instances of the right-hand sides. Thus if $x*(y+z) \rightarrow x*y+x*z$ is a rewrite rule, then using this rule we can rewrite the expression $a*(b+c)+2$ to $(a*b+a*c)+2$. The questions of interest are, when does this rewriting operation terminate, when is it confluent (that is, the result of a sequence of rewrites does not depend on the choice of how the rules are applied, roughly speaking), and when can rewriting be used to decide the corresponding equational theory, among others. There are obvious connections to automated deduction, functional programming, and algebraic specification. Indeed, theorem provers based on the theory of term-rewriting systems are particularly impressive. Some algebraic specification systems use term-rewriting systems as their operational semantics. Also, the connection of term-rewriting with functional and logic programming is being continually explored.

The papers in this special issue cover a number of areas of term-rewriting systems. Beginning with the papers nearest to the center of the field, the four papers of Bernhard Gramlich, Ursula Martin, Joachim Steinbach, and H. Zantema all concern termination, in one way or another. Termination is in general undecidable, so special methods have to be developed to demonstrate termination in special cases of interest. Gramlich's paper demonstrates relationships between restricted termination properties and confluence. Ursula Martin's paper reveals connections with the theory of ordinals, and goes on to classify certain orderings used to prove termination. Joachim Steinbach gives a survey of termination, and H. Zantema gives a method of proving termination that is based on semantic, rather than syntactic, arguments. Moving on now to the other papers, Hantao Zhang discusses a generalized version of rewriting that is related to conditional term-rewriting systems; these are rewrite rules such as $(s \to t)$ *if* C that may only be applied if the condition C is satisfied. The paper by Bockmayr, Krischer, and Werner is about *narrowing*; this is a version of rewriting in which one seeks an instance t' of some initial term t such that t' rwerites to a specified value (such as *true*). Narrowing is of interest because it provides a way to unite term-rewriting and logic programming, in that the instance t' has variables instantiated to terms as in a Prolog answer substitution. Paliath Narendran explores a unification problem and its relationship with a satisfiability problem involving Horn clauses. Unification and rewriting are closely connected via equational rewriting. The paper by Gilleron and Tison explores a connection between term-rewriting systems and regular tree languages, which are in turn a generalization of the traditional regular lanuages of automata theory, to trees. Finally, the paper by Bonacina and Hsiang explores a method for parallel implementation of theorem provers, and applies it to some theorems that can be expressed as sets of equations, obtaining in this way impressive speedups.

We hope that this selection of papers will provide an inviting introduction to some of the areas of term-rewriting systems research and will encourage the readers to develop an increased familiarity with this beautiful, applicable, and fast-growing field of study.