Expression Reduction Systems and Extensions: An Overview

John Glauert[1], Delia Kesner[2], and Zurab Khasidashvili[3]

[1] School of Computing Sciences, University of East Anglia, Norwich, UK
J.Glauert@uea.ac.uk
[2] PPS, CNRS and Université Paris 7, France
kesner@pps.jussieu.fr
[3] Logic and Validation Technology,
Design Technology Division Intel Development Center, Haifa, Israel
zurabk@iil.intel.com

Abstract. Expression Reduction Systems is a formalism for higher-order rewriting, extending Term Rewriting Systems and the lambda-calculus. Here we give an overview of results in the literature concerning ERSs. We review confluence, normalization and perpetuality results for orthogonal ERSs. Some of these results are extended to orthogonal conditional ERSs. Further, ERSs with patterns are introduced and their confluence is discussed. Finally, higher-order rewriting is translated into equational first-order rewriting. The technique develops an isomorphic model of ERSs with variable names, based on de Bruijn indices.

1 Introduction

Many programming languages and logical systems are modelled by transformations that allow programs or expressions to be rewritten as values that are just simpler expressions.

The more traditional rewriting frameworks are *first-order rewriting systems*, where expressions are represented using term algebras, and the λ-*calculus*, where expressions are modelled by λ-terms. Both languages can be combined naturally into a *higher-order rewriting system*, which is a well-suited formalism to deal simultaneously with algebraic data structures and functions. We thus obtain a framework inheriting the advantages from both the first-order and the functional worlds.

Following the ground-breaking work by J. W. Klop [67] on *Combinatory Reduction Systems* (CRS), many different frameworks for higher-order rewriting have been proposed: *Expression Reduction Systems* (ERS) by Z. Khasidashvili [52], *Higher-Order Rewrite Systems* (HRS) by T. Nipkow [81], *Higher-Order Rewriting Systems* (HORS) by V. van Oostrom and F. van Raamsdonk [98], *Higher-Order Term Rewriting Systems* (HOTRS) by D. Wolfram [101], *The General Scheme family* (GS) or *Algebraic-Functional Systems* (AFS) by Jouannaud and Okada [46].

A. Middeldorp et al. (Eds.): Processes... (Klop Festschrift), LNCS 3838, pp. 496–553, 2005.

In this chapter we review some results concerning *Expression Reduction Systems*. We start this introduction by relating ERS to Pkhakadze's work as well as to other well-known formalisms in the literature.

1.1 A Short History of ERSs

Expression Reduction Systems (ERSs) were introduced by Khasidashvili [52], under the supervision of Sh. Pkhakadze. The syntax of ERSs was influenced by the syntax of the Notation Theory of Pkhakadze [86, 87].

Pkhakadze's work was motivated by a study of formal mathematical theories and, in particular, extensions of formal theories with new function or predicate symbols. When new symbols are *defined* using the original or previously defined symbols, one expects that the new, extended theory is a conservative extension of the original one – no new results can be proven on expressions of the original theory.

Besides function and predicate symbols, one often needs to introduce new quantifiers. Bourbaki [18] devoted a large part of their study of set theory to the introduction of new symbols in formal theories. However, a general syntactic framework for defining new symbols was missing there, which would enable, for all introduced symbols, uniform proofs of syntactic results, such as termination or confluence in current rewriting terminology. The aim of Pkhakadze's work [86] was indeed to define a uniform syntax for defining new symbols, and the hard work went into understanding the binding structure of the rules defining new quantifiers. A famous example of a definition of a quantifier is Hilbert's definition of the existential quantifier using the choice operator τ: $\exists x(A) \to (\tau x(A)/x)A$.

Pkhakadze introduced several syntactic categories for defining new symbols. The most general of them are of the form $\sigma a_1 \dots a_n(A_1 \dots A_m) \to B$, where a_i are object-metavariables expressing, after instantiation, the binding variables of the quantifier symbol σ. The A_j are term-metavariables that are instantiated to terms or formulae, and B is a meta-expression written using metavariables occurring in the left-hand side and using meta-substitutions of the form $(A_l/a_k)A_r$. Clearly, for the rewrite relation to be well defined, several syntactic constraints need to be imposed on the right-hand sides of the rules, which was done in [86]. These conditions are described in detail in a more recent summary of Pkhakadze's work (written in English) [87].

It would be fair to say that the alphabet in Pkhakadze's system was typed – there were symbols that could take terms or formulae as arguments, and return terms or formulae, depending on their types. The early versions of ERSs [52] used a similar syntax – symbols of types NAT and $BOOL$ were used, but we cannot see them in later versions of ERSs. Similarly, object-metavariables are no longer used in rewrite rules of ERSs.

The main results in [86] concern termination of rewriting (or elimination of defined symbols) and uniqueness of the normal forms. Most of the results there correspond to first-order rewriting. For more information, we refer to [87].

While ERSs were introduced independently from Klop's work [67], later work of the third author on ERSs, and especially on perpetual strategies, was greatly