ABSTRACT REWRITING SYSTEMS AND SUBSYSTEMS: UNDESTANDING REWRITING STRATEGIES AND SUB-CALCULI

A. Arbiser (Buenos Aires)

In rewriting theory, an abstract rewriting system (ARS) is a pair $(A, \to)$ where A is a set (whose objects we call terms) and $\to$ is a binary relation on A. For any relation $\to$, we denote with $\overset{*}{\to}$ its reflexive-transitive closure. A sub-ARS of a given ARS is an ARS whose set of terms is a subset of the original and closed by reduction and with its relation being the restriction of the original ARSs relation to the latter. Sub-ARSs always inherit the rewriting-related properties of the original ARS (such as being confluent, weakly confluent, terminating, finitely branching, etc.) We intend to study sub-ARSs for term rewriting systems (TRSs, rewriting over first-order signatures), context-sensitive rewriting systems (CSRSs, where some function symbols do not allow rewriting under specific argument positions), semi-Thue systems (STSs, rewriting over strings using pairs of strings as rules), and different forms of lambda calculus; and to investigate a number of properties and relations between ARSs and sub-ARSs for these rewriting formalisms.

ARSs are the objects of a cartesian category, where the sub-ARSs are the sub-objects, and they form a complete lattice. Little or no work has been done in studying sub-ARSs, ARSs products and ARSs of functions.

Since rewriting systems are essentially non-deterministic, there is interest in rewriting strategies. Formally, and following a definition of Van Oostrom and de Vrijer, a reduction strategy $\to_s$ for a given ARS is any sub-ARS preserving the normal forms. Intuitively, a strategy is a criterion on which to select a rewrite step over the existing ones (this is equivalent to the other definition). If $\to_s$ is a function, we call it a functional (or deterministic) reduction strategy.

A reduction strategy $\to_s$ is complete with respect to subset B of terms if, given any term a, if $a \overset{*}{\to} b$ with $b \in B$, then $a \overset{*}{\to}_s b$. For instance, it is known that in the usual lambda calculus the leftmost-outermost reduction strategy is complete with respect to the set of normal forms. We investigate some sets where (some) strategies are complete.

Given an ARS one would like to generate all its terms minimally. In this sense, we call base to any minimal subset B of terms such that for every term b, there exists $a \in B$ such that $a \overset{*}{\to} b$. We exhibit specific examples of sub-ARSs of the lambda calculus, and we show that lambda calculus (and other reasonable rewriting systems) do not admit bases, finite nor infinite. So, even though for any ARS the axiom of choice enables to have a minimal set B with the property that each term is equivalent (modulo reduction) to a unique term in B, the situation changes if arrow direction is considered.

There are relations between the rewriting formalisms. Every STS is isomorphic to a linear TRS with only unary symbols. Every ground TRS (i.e. without variables) is isomorphic to a sub-ARS of an STS. We prove that there are linear TRSs which are not isomorphic to any sub-ARS of any STS, but every TRS is isomorphic to a sub-ARS of an augmented STS (where rules can have variables denoting substrings). And, since there are ARSs which are not isomorphic to any TRS, the tower of formalisms STS $\subset$ TRS $\subset$ ARS is proper. Every n-sorted TRS is isomorphic to a sub-ARS of a one-sorted TRS which uses only one constant and one binary symbol. Furthermore, given an ARS $(A, \rightarrow)$, new non-trivial ARSs can be formed with the set of finite A-subsets, having $(A, \rightarrow)$ as one sub-ARS, and preserving some of the rewriting properties of $(A, \rightarrow)$: confluence, weak and strong normalization, diamond property, etc. This has some relation with residual aspects of rewriting.

We are also interested in the implementation of various rewriting systems. At present there is a big number lambda calculus variations, in particular lambda calculi with explicit substitutions (such as $\lambda\sigma$, λs, $\lambda\upsilon$). Explicit substitutions constitute a way of implementing lambda calculus. Since calculi of explicit substitution with de Bruijn indices are TRSs in some sense, we implemented a code generator which (more generally) transforms a given CSRS to a Haskell program implementing some specific strategy (so each function symbol can be evaluated in an outermost or innermost way, and also the contextual rewriting can be restricted to a given subset of its arguments). The idea of the implementation is to find (and experiment with) appropriate strategies for (possibly context-sensitive) lambda calculi with explicit substitution, where the strategy may depend on each function symbol. For example, the $\lambda\upsilon$-calculus (cf. Lescanne et al.) with open terms can be identified to a sub-ARS of the one-sorted TRS with rules: $\{\ ap(la(x), y) \rightarrow cl(x, sl(y))$, $cl(ap(x, y), s) \rightarrow ap(cl(x, s), cl(y, s))$, $cl(la(x), s) \rightarrow la(cl(x, li(s)))$, $cl(one, sl(x)) \rightarrow x$, $cl(suc(n), sl(x)) \rightarrow n$, $cl(one, li(s)) \rightarrow one$, $cl(suc(n), li(s)) \rightarrow cl(cl(n, s), sh)$, $cl(n, sh) \rightarrow suc(n)\ \}$, where x, y, s, n are variables. Moreover, from the above rules we can obtain Haskell code implementing leftmost-outermost, leftmost-innermost, and other reduction strategies, given by the usual two-sorted (terms and substitutions) algebra or as a sub-calculus of a one-sorted algebra.

This is work in progress and there are many research directions. Although for the simplistic one-sorted two-symbol representations of a calculus there are many choices, we are interested in determining natural or economic candidates. It could be interesting to explore which known properties of a calculus are valid for the full TRS (apart from the sub-ARS isomorphic to the calculus). Also, we look for other characterizations of sub-ARSs of different rewriting systems, the completeness of strategies and the study of history-dependent strategies.

References

[1] M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit substitutions. *Journal of Functional Programming*, 4(1), 1991.

[2] F. Baader, T. Nipkow. Rewriting and all that. Cambridge Univ Press, 1999.

[3] H.P. Barendregt. The Lambda Calculus: its Syntax and Semantics. Studies in Logic and the Foundations of Mathematics 103. North-Holland, Amsterdam, revised edition, 1984.

[4] R. Bloo. Preservation of Termination for Explicit Substitutions. PhD thesis, Eindhoven University, 1997.

[5] A. Church and J.B. Rosser. Some properties of conversion. *Transactions of the American Mathematical Society*, 39, 1936.

[6] Th. Hardin, J.-J. Levy. A confluent calculus of substitutions. In France-Japan Artificial Intelligence and Computer Science Symposium, 1989.

[7] J.W. Klop. Combinatory Reduction Systems. PhD thesis, Mathematical Centre Tracts n.127, CWI, 1980.

[8] P. Lescanne and J. Rouyer-Degli The Calculus of Explicit Substitutions λv. CNRS and INRIA-Lorraine, 1995.

[9] S. Lucas. Context-sensitive rewriting, lazy rewriting, and on-demand rewriting DSIC, Univ. Pol. Valencia, 2002.

[10] P-A. Melliès. Description Abstraite des Systèmes de Réécriture. PhD thesis, Université Paris VII, 1996.

[11] V. van Oostrom, R. de Vrijer. Equivalence of reduction strategies. TR 2001.

[12] F. van Raamsdonk. Confluence and normalization for higher-order rewriting. PhD thesis, Amsterdam University, 1996.

[13] A. Rios. Contributions a l'etude des lambda-calculs avec substitutions explicites. These de Doctorat, Universite Paris VII, 1993.