# Modular Context-Sensitive Algebraic Specifications

Bernhard Gramlich[1⋆] and Salvador Lucas[2⋆⋆]

[1] Fakultät für Informatik, Technische Universität Wien
Favoritenstr. 9, A-1040 Wien, Austria
email: gramlich@logic.at, www: www.logic.at/staff/gramlich/

[2] DSIC, Universidad Politécnica de Valencia
Camino de Vera s/n, 46022 Valencia, Spain
email: slucas@dsic.upv.es, www: www.dsic.upv.es/users/elp/slucas.html

**Abstract** *Context-sensitivity* in algebraic specifications is an interesting approach for bridging the gap between the purely logical and semantic nature of such systems and (certain problems in) their operational realization and implementation. *Context-sensitive* algebraic specifications are obtained by enriching ordinary algebraic specifications with additional syntactical (context) information that turns out to be very useful in transforming these specifications into executable prototypes or implementations satisfying usually desired properties. Here we investigate modular aspects of such context-sensitive algebraic specifications and computations in the framework of *context-sensitive equational reasoning* and *rewriting*. In particular, we study the modularity behaviour of context-sensitive term rewriting systems w.r.t. confluence and related uniqueness properties. We show how to extend various modularity results known from (unrestricted) term rewriting systems to this more general setting, and how to cope with the additional complications caused by context-sensitivity.

**Background** Context-sensitive rewriting, CSR for short (cf. e.g. [8, 9]) has recently emerged as an interesting, powerful and flexible paradigm that provides a bridge between the abstract world of general algebraic specification and computation (rewriting) and the (more) applied setting of declarative specification and programming languages such as the OBJ-family, ELAN, and Maude. A natural approach to study properties of specifications and programs written in these languages is to model them as context-sensitive (equational and) rewriting systems (CSRSs for short). The basic idea of context-sensitive rewriting is very simple: Computing, i.e., reducing, some subterm in a given term is only possible if this is (recursively) allowed in the *context* of the subterm. This is modelled by specifying for every function symbol $f$ which of its argument positions are *active* (or *replacing*) and which ones are not (i.e., *inactive* or *forbidden*). On the equational and semantical level, context-sensitivity means that equality is no longer a full

⋆ Work partially supported by ÖAD-Programm "Acciones Integradas 2002-2003", under grant No. 3/2002

⋆⋆ Work partially supported by MCyT TIC2001-2705-C03-01, Acción Integrada HU 2003-0003, and Agencia Valenciana de Ciencia y Tecnología, under grant GR03/025

congruence, i.e., monotonicity (w.r.t. contexts) is partially lost. As a basic simple example consider the typical case of a ternary *if*-operator specified by the two rules $if(true, \boxed{s}, \boxed{t}) \to s$, $if(false, \boxed{s}, \boxed{t}) \to t$, where the boxes indicate that the corresponding arguments of *if* must **not** be evaluated. In other words, here only the condition of an *if*-expression may be further evaluated. Another basic example are *lazy infinite* lists generated e.g. by $inf(x) \to cons(x, \boxed{inf(s(x))})$ where $s$ is the successor function for natural numbers. Here the second argument of *cons* is declared to be *forbidden*, thus avoiding infinite expansions. But note that whenever by means of other rules (for instance for selecting a given number of elements from a list) the second argument $inf(s(x))$ of the right-hand side above happens to migrate into an active position of some term to be evaluated, then further computations become possible. Hence, laziness, i.e., constructing only parts of the infinite data structure *by need*, is triggered by appropriate context information.

This kind of context-sensitivity restrictions has turned out to be very useful for obtaining better computational properties, e.g., for increased efficiency, a better termination behaviour, and an effective handling of infinite data structures. On the other hand, context-sensitivity significantly complicates the theoretical analysis of such specifications and computations, due to its non-monotonic nature. Much of the well-known theory in term rewriting does not carry over anymore easily to this new setting. Context-sensitivity in the above sense goes beyond ordinary algebraic specifications and is not explicitly addressed in the *Common Framework Inititive (CoFI)*[3] and in corresponding language approaches like *CASL* (cf. [10]). Yet, context-sensitivity has recently been explicitly included in similar theoretical frameworks, cf. e.g. the *Generalized Rewrite Theories* of [2]. As argued above above, we think that this concept is well-suited to provide structured and high-level support for the analysis and operationalization (prototype construction and implementation) of algebraic specifications (as in CASL) in such a way that essential properties are preserved or enforced to hold.

**State-of-the-Art** Meanwhile the logical, semantical and computational properties of context-sensitive algebraic specifications and rewrite systems have been studied and understood to some extent (cf. e.g. [9]). This concerns especially methods, techniques and criteria for verifying termination and confluence properties of CSRSs. Regarding the termination property, a considerable amount of work already exists, where via tranformational approaches termination of some CSRS is reduced to termination of some transformed (more complicated) ordinary term rewriting system (TRS for short), cf. eg [3]. [4] However, these transformational approaches have some serious drawbacks, in the sense that the translations are in general neither structure preserving nor compatible with modularity and compositionality needs. Especially the latter aspect is crucial. A modular design of big systems as well as a modular analysis of large specifications and programs (defined by rewrite systems) is vital in practice (cf. e.g.

[3] see `http://www.cofi.info/`

[4] For a recent direct *rpo*-style approach to prove termination of CSRSs we refer to [1].

[11], [4]). Motivated by this need, we have recently initiated some first promising steps towards a thorough modularity analysis of CSR. In [5, 6] we have analysed whether various modularity results for termination of TRSs carry over to the context-sensitive case, and if so, under what (additional) conditions. These results have been very promising, but also revealed some subtleties and previously unknown complications.

**Our Contribution** In this work, we continue the modularity analysis of [5, 6], but rather focus on confluence and related uniqueness properties. When one wants to guarantee that results of context-sensitive computations are (in some reasonable sense) unique, then such a modularity analysis yielding sufficient syntactically verifiable criteria for confluence properties of composed larger systems is indispensable. We first deal with the case of disjoint unions of CSRSs and show how to extend known results from the TRS case (cf. e.g. [11, 7]) to CSRSs. Subsequently we also address certain cases of non-disjoint unions of CSRSs like combinations *sharing (at most) constructors.*

## References


1. C. Borralleras, S. Lucas, and A. Rubio. Recursive path orderings can be context-sensitive. In A. Voronkov, ed., *Proc. 19th Int. Conf. on Automated Deduction (CADE'02), Copenhagen, Denmark*, LNCS 2392, pp. 314–331. Springer, July 2002.
2. R. Bruni and J. Meseguer. Generalized rewrite theories. In J. Baeten et al., eds., *Proc. 30th Int. Conf. on Automata, Languages and Programming (ICALP'03), Eindhoven, The Netherlands*, LNCS 2719, pp. 252–266, Springer, June/July 2003.
3. J. Giesl and A. Middeldorp. Transformation techniques for context-sensitive rewrite systems. *Journal of Functional Programming*, 2004. To appear, 49 pages.
4. B. Gramlich. Modular aspects of rewrite-based specifications. In F. Parisi-Presicce, ed., *Recent Trends in Algebraic Development Techniques, 12th Int. Workshop, WADT 97, Tarquinia, Italy, June 1997, Selected Papers*, LNCS 1376, pp. 253–268. Springer, 1998.
5. B. Gramlich and S. Lucas. Modular termination of context-sensitive rewriting. In *Proc. 4th Int. Conf. on Principles and Practice of Declarative Programming (PPDP 2002)*, pp. 50–61, Pittsburgh, PA, USA, Oct. 2002. ACM Press.
6. B. Gramlich and S. Lucas. Simple termination of context-sensitive rewriting. In B. Fischer and E. Visser, eds., *Proc. 3rd ACM Sigplan Workshop on Rule-based Programming (RULE'02)*, pp. 29–41, Pittsburgh, PA, USA, ACM Press, Oct. 2002.
7. J. W. Klop, A. Middeldorp, Y. Toyama, and R. Vrijer. Modularity of confluence: A simplified proof. *Information Processing Letters*, 49:101–109, 1994.
8. S. Lucas. Context-sensitive computations in functional and functional logic programs. *Journal of Functional and Logic Programming*, 1998(1), 1998. The MIT Press.
9. S. Lucas. Context-sensitive rewriting strategies. *Information and Computation*, 178(1):294–343, 2002.
10. P. Mosses, ed. *CASL, The Common Algebraic Specification Language – User Manual and Reference Manual, by CoFI, The Common Framework Initiative for Algebraic Specification and Development.* IFIP Series. Springer, 2003. To appear.
11. Y. Toyama. On the Church-Rosser property for the direct sum of term rewriting systems. *Journal of the ACM*, 34(1):128–143, 1987.