

Book review

submitted 14 March 2005; revised 23 March 2005; accepted 24 March 2005

Term Rewriting Systems by “Terese” (Marc Bezem, Jan Willem Klop, and Roel de Vrijer, eds.), Cambridge University Press, *Cambridge Tracts in Theoretical Computer Science* **55**, 2003, hard cover: ISBN 0-521-39115-6, xxii + 884 pages.

The formal study of rewriting and its properties began in 1910 with the pioneering work of Axel Thue, and has since developed into a major area of research. Broadly speaking, *rewriting* is the study of normal forms, their existence, their uniqueness, and their computation. As a discipline of computer science, term rewriting has two main application areas: functional languages and their semantics; and equational reasoning and mechanical inference. *Term Rewriting Systems* provides a very comprehensive treatment of the subject, mainly from the former point of view.

The rhetorical question, “Who is Terese?” the ostensible author of the volume, is answered in the prefatory pages. “Terese” is the pseudonym of the contributing editors, Marc Bezem, Jan Willem Klop, and Roel de Vrijer, and their colleagues, Erik Barendsen, Inge Bethke, Jan Heering, Richard Kennaway, Paul Klint, Vincent van Oostrom, Femke van Raamsdonk, Fer-Jan de Vries, and Hans Zantema, who all shared in the writing of this massive work. The authors were all, at one time or another, members of the TeReSe (“*Term Rewriting Seminar*”) group, which held regular meetings during the years 1988–2000 at the Vrije University in Amsterdam.

The genesis of this book went something like this: In the beginning, in 1985 C.E., Klop delivered a short course on term rewriting at a seminar in Naples. He reworked the unpublished course notes into a tutorial for a 1987 issue of the *Bulletin of the European Association of Theoretical Computer Science*. Then, in 1990, he gave a lecture on rewriting at the International Colloquium on Automata, Languages and Programming, held at Warwick University. This grew into a large chapter in the *Handbook of Logic in Computer Science* (Oxford Univ. Press, 1992), which was the twelfth most frequently cited paper in CiteSeer’s gigantic database of online papers, circa 2000–2001, and remains high up on their list. Determined to expand this material into a full-length book, the Terese team of contributors was assembled.

Terese labored many years to bestow upon the scientific community a large and appealing volume, comprising sixteen chapters, plus an appendix. This is not a gentle introduction to the field (there are handbook chapters for that), but a text that is both broad and deep. One or more of the editors authored or coauthored ten of the chapters. Some are virtually full-length monographs; most are comprehensive discourses; a few are bare-bones sketches. This is the first book to provide rigorous coverage of topics such as standardization, neededness, and infinitary rewriting.

Algorithmic issues are played down. The chapters are complemented by an encyclopedic bibliography of some 500 references, and an index.

Term Rewriting Systems starts off with an introductory Chapter 0, presenting three examples of rewriting, well-chosen for their heterogeneity: a functional program for integer division; Reidemeister moves for unravelling knots; and Jieh Hsiang’s nondeterministic rules for computing the Zhegalkin (Boolean-ring) normal form of a propositional formula.

As has become customary, as many of the basic concepts as possible are introduced within the abstract framework of arbitrary binary relations. Thus, Chapter 1, by Bezem and Klop, defines (and fixes notations for) the fundamental notions of confluence, including the Church-Rosser property (CR), and of normalization—in both its weak (WN, that is, existence of normal forms) and strong (SN, or, uniform termination) forms. Then it goes on to enumerate the relations between the notions, most notably the 1942 result of Max Newman that local confluence (WCR) and strong normalization imply the Church-Rosser property. (Three proofs of Newman’s Lemma are given *ad loc.* and one more in Chapter 14.)

Chapter 2, by Klop and de Vrijer, turns to the central theme of the volume: rewriting first-order terms. After defining the basic notions of term, context, substitution, (syntactic) unification, reduction (i.e. rewriting), and many more, it uses artful diagrams to explain the concepts of overlap and critical pair, and prove Don Knuth’s famous Critical Pair Lemma.

Two important, early examples of term rewriting, namely Herbrand and Gödel’s general recursive functions and Curry’s Combinatory Logic, are explained in Chapter 3. These are followed by short sections on ground rewriting, recursive program schemes, many-sorted rewriting, string rewriting (a.k.a. semi-Thue systems), conditional rewriting (with equational Horn clauses), and the $\lambda\sigma$ -calculus (the earliest of numerous recent proposals of a lambda calculus with explicit operations for performing substitutions).

The pivotal chapter is the fourth. In it, Klop, van Oostrom, and de Vrijer define *orthogonal* (left-linear and overlap-free) term rewriting systems and prove (again, in more than one fashion) that they are confluent, hence, have at most one normal form per term. The core ideas of descendants, residuals, developments, and reduction diagrams are clearly explained, and the central issue of normalizing strategy is introduced. (Regrettably, most of the proofs about normalization are deferred to a later chapter.)

Chapter 5, by Klop and de Vrijer, is in reality two chapters: The first proves that most interesting properties of term rewriting are—no surprise here—undecidable. The second part gathers many results on the question of the extent to which the union of two systems that share no function symbols or constants, but share a property like confluence or strong normalization, also enjoys that property. The two central results are a theorem by Yoshihito Toyama to the effect that confluence is preserved, proved in detail, and another that confluence plus strong normalization are preserved for left-linear systems, which is stated without proof. The book does not deal with cases in which constructors or other symbols are shared (early on recognized as important for practical modularity of programming considerations).

The termination chapter, by Zantema, is a long, somewhat haphazard survey of syntactic, semantic, and transformation-based methods of proving strong normalization (termination), including the recent dependency-pair method (*sans* proof).

Chapter 7, by Bethke, contains a very brief treatment—only 40 pages long—of equational inference, including the all-important Knuth-Bendix completion procedure, presented in the style of inference rules and proof orderings (as developed by Leo Bachmair), proof by consistency (invented by Dave Musser, who should have been cited), and equation solving (semantic E -unification). Many topics are not addressed, including “unfailing” completion and associative-commutative completion, despite their importance for automated deduction. Also, the contributions of Gérard Huet to our understanding of the rôle of completion as theorem prover and of Jieh Hsiang to the use of Boolean rings for first-order reasoning go unmentioned.

Chapters 8 and 9, both by van Oostrom and de Vrijer, comprise a very extensive and beautiful study (at almost 250 pages, nearly one third of the text!) of properties of derivations of left-linear systems and reduction strategies for orthogonal systems, including much new material. Labelling methods, tracing methods, various forms of reduction (derivation) equivalence, standardization, reduction strategies, sequentiality, “family” rewriting, and much more are covered in great depth.

Two digressions are included next: the basics of untyped and typed lambda calculi in a chapter by Bethke, followed by an introduction to higher-order rewriting and combinatory reduction systems by van Raamsdonk. Of course, full treatment of either would require a whole book unto itself. Still, they serve nicely to link rewriting to sibling fields.

Properties of infinitely-long reduction sequences for orthogonal systems are the subject of Chapter 12 by Kennaway and de Vries. It is refreshing to see this relatively new topic in a textbook. The less interesting, and more problematic, case of Cauchy-converging sequences is not elaborated.

Term-graph rewriting, important for implementations, is briefly discussed by Barendsen in Chapter 13. Some advanced topics, like de Bruijn and van Oostrom’s method of decreasing diagrams, are presented in Chapter 14 by Bezem, Klop, and van Oostrom. Chapter 15, by Heering and Klint, is a welcome, though incomplete, list of twenty-seven implementations of varied flavors of rewriting.

Last, but by no means least, the appendix by Bezem contains a masterful review of mathematical prerequisites, including Cantor’s ordinal numbers, Kruskal’s Tree Theorem, and Ramsey’s Theorem. The book closes with a pretty exercise on the evolution of “amoebæ colonies.”

In fact, all chapters (save, naturally, the first and last) include exercises, some more generously, and others less so. Keeping up with the times, the editors maintain a web site, www.cs.vu.nl/~terese, with solutions to many exercises (but, so far, only for half the chapters).

The editors are also to be commended for what was obviously a gargantuan effort expended in maintaining uniformity of form across the writings of a dozen people. Abundant use is made of a few abbreviations: “ARS” for “abstract reduction system” (already used on p. 7, just prior to its definition); “TRS” for “term rewriting system,” “HRS” for higher order, etc. As of January 2005, the web site

lists four errata; there are relatively few typos. Cambridge should have provided better copyediting: though always clear, the English is occasionally stilted. The notation is good overall, despite some minor inconsistencies of usage. For example, the letter \mathcal{R} is used for rewriting systems (signature plus rules) in Chapters 2–5, for reductions in 8–9, and for sets of redexes in 12.

The copious references given in all chapters are very helpful, though, here and there, there are lapses. I think I have decoded the non-standard order of the references; I leave that as a challenging exercise to the reader! The name and subject indices are quite comprehensive and useful, but a bit inconsistent in coverage. For example, Newman’s Lemma and Toyama’s Theorem (about which much ado is made in the introduction) are not indexed. The notation index is harder to utilize, and a few symbols are missing from the list. All in all, my students found the cross-references between chapters very handy, and the examples well-placed and helpful. They were particularly happy with the multiple insights, perspectives, and proof methods afforded throughout.

For an introductory course in rewriting, I would choose Chapter 0, as much as is needed from Appendix A, Chapters 1, 2 and 4, Sections 9.3, 5.1–3, 6.1–2, and 6.4, and Chapter 7. For an in-depth treatment of orthogonal rewriting, one could choose Chapters 1, 2, 4, 8, and 9, with Chapters 10 and 12 as optional extras. Anyone interested in a more exhaustive discourse on equational theorem proving will need to supplement Chapter 7 with additional material. For example, the older book, *Term Rewriting and All That*, by Franz Baader and Tobias Nipkow (Cambridge University Press, 1998), has a more detailed discussion of associative-commutative unification, but also only a brief account of unifying and associative-commutative completion. For students of logic programming, the book, *Advanced Topics in Term Rewriting*, by Enno Ohlebusch, reviewed in these pages (vol. 4, pp. 539–541), has extensive material on conditional rewriting and termination of logic programs, which could be of interest.

Suffice it to say that this book is indispensable for any serious student of rewriting.

NACHUM DERSHOWITZ
Tel Aviv University, Tel Aviv