Rewriting in Practice*

Ashish Tiwari[1]

1 SRI International
Menlo Park, CA 94025
tiwari@csl.sri.com

Abstract

We discuss applications of rewriting in three different areas: design and analysis of algorithms, theorem proving and term rewriting, and modeling and analysis of biological processes.

1998 ACM Subject Classification F.4.2 [Mathematical Logic and Formal Languages] Grammars and Other Rewriting Systems–Decision problems; I.6.5 [Simulation and Modeling] Model Development–Modeling Methodologies; I.1.2 [Symbolic and Algebraic Manipulation] Algorithms–Algebraic Algorithms

Keywords and phrases Rewriting, Polynomial constraints, Biochemical reaction networks

Digital Object Identifier 10.4230/LIPIcs.RTA.2011.3

Category Invited Talk

1 Introduction

The field of rewriting has contributed some fundamental results within the computer science discipline. Here, we explore a few impactful applications of rewriting. Any article that describes applications of a field has to be necessarily incomplete. We limit our focus on the use of rewriting technology in the following three areas.

1. Design of algorithms
2. Formal modeling and analysis
3. Term rewriting and theorem proving

We discuss the influence of the theory of rewriting in the above areas by identifying specific concrete instances of algorithms, tools, or techniques that have, or can, be impacted by rewriting.

The field of rewriting is broadly concerned with manipulating *representations* of *objects* so that we go from a larger representation to a *smaller* representation. Clearly, rewriting is concerned with three important entities: objects, representations, and orderings. We give a few examples below.

- In term rewriting, the objects are equivalence classes of terms, representations of these objects are the terms themselves, and orderings are certain binary relations on terms.
- In polynomial rings, objects are polynomials and representations are algebraic expressions constructed using the arithmetic (ring) operators.
- In theorem proving, objects are proofs and orderings are proof orderings on some representation of the proofs.

A significant part of the theory of rewriting abstracts away from the objects and/or their representations, and just studies properties of binary relations.

* Research supported in part by the National Science Foundation under grant CSR-EHCS-0834810 and CSR-0917398.

2 Rewriting in the design of algorithms

The theory of rewriting can be used as a basis to study the design and analysis of algorithms. There are at least two distinct ways in which rewriting helps in algorithmic development.

2.1 Rewrite-based Descriptions

An algorithm can often be viewed as a set of rewrite rules. This *rewrite-based description* cleanly separates the logical part of an algorithm from its implementation details. Rewrite-based views of algorithms can be very helpful especially from an educational perspective.

Consider, for example, the problem of sorting a sequence of numbers, which is one of the most commonly studied problems in a first course in Algorithms. A description of a sorting procedure can be given by a single rewrite rule

$$X, a, Y, b, Z \quad \to \quad X, b, Y, a, Z \qquad \text{if } a > b$$

where , is an associative operator. This rewrite rule simply says that we can swap two elements in the current list if they are not in order. This rewrite rule describes a whole class of comparison-based sorting algorithms: different concrete sorting algorithms, such as bubble sort, are obtained by applying (specific instances of) the above rewrite rule using different *strategies*. The correctness – soundness, completeness, and termination – of comparison-based sorting algorithms just depends on properties of the rewrite system containing the above rule. The strategy (and the data structures used to represent the terms) is only an implementation detail (although an important detail since it determines the final time and space complexity). Graph algorithms can be similarly presented abstractly using rewrite rules.

2.2 General Paradigm

Rewriting provides a general paradigm for the design of algorithms. The (abstract) critical-pair completion algorithm is a generic procedure that can be instantiated in different domains to yield very important algorithms, such as,

- the algorithms implementing the union-find data structure [9],
- congruence closure [24, 8] and associate-commutative congruence closure [6],
- Gröbner basis algorithm [11, 3, 7], and
- Simplex algorithm for satisfiability of linear constraints [14, 15].

The completion-based rewriting view of these algorithms yield simpler proofs of correctness of these complex algorithms.

There is, however, an additional benefit of taking the completion-based view. It becomes possible to inherit certain optimizations. Consider the Gröbner basis algorithm. It involves adding equations arising from critical overlap of polynomial equations. If we have the following two equations in the current set of equations

$$x^2 \overset{(1)}{\to} p \qquad\qquad\qquad xy \overset{(2)}{\to} q$$

where x^2 is the maximal monomial in the polynomial $x^2 - p$ and xy is the maximal monomial in the polynomial $xy - q$ (assume some ordering, say a total degree lexicographic ordering with precedence $x \succ y$), then the procedure for computing Gröbner basis adds the new equation $py \overset{(3)}{=} qx$ (arising from the cricial pair between the two rules above) to the current set of polynomial equations. Efficiency of Gröbner basis algorithms is determined by the number of such critical equations generated. Hence, deletion is important: if we can delete an equation or rule, we compute fewer inferences subsequently. In the above example, we note that we cannot delete any of the two original rules. However, we can delete the instances $x^2yX \to pyX$ of the first equation, where X is an arbitrary power product. The reason why we can delete these instances is that they all have a proof that does not use the first rule

$$x^2yX \overset{(2)}{\to} qxX \overset{(3)}{=} pyX$$

Moreover, in a suitably defined proof ordering, this new proof of $x^2yX = pyX$ can be shown to be smaller than the old proof that uses Rule (1). Thus, in the new optimized Gröbner basis procedure, rules are associated with a list of monomials called `forbidden`, with the interpretation that instances of a rule obtained by multiplying that rule with a monomial in the ideal generated by `forbidden` are assumed to have been deleted. This optimization has been mentioned before [1, 33], and it appears to be related to some of the new *signature-based* algorithms for Gröbner basis [16]. An exciting future work would be to study the signature-based algorithms using the rewriting framework.

3 Rewriting in Term Rewriting and Theorem Proving

The essence of the theory of rewriting, and in particular of the critical-pair completion procedure, can be described as

1. add facts that make proofs of provable facts smaller and
2. delete facts that already have smaller proofs

Here, by facts we mean equations, or clauses, or formulas (ground or non-ground), or any representation of the known object of interest. This more general view of completion inspires the design of several other algorithms; most notably the algorithms used in saturation-based theorem proving [4].

In equational reasoning, proofs have a certain nice structure that enables one to define interesting proof orderings, which leads to algorithms such as standard and ordered completion [2]. When considering non-equational theories, proofs do not necessarily have a very nice structure. So, complexity of a proof is often just the complexity of the facts used in the proof. Consider the resolution inference rule, where given the two (propositional) clauses

$$x \;\vee\; C_1 \qquad\qquad \neg x \;\vee\; C_2$$

the resolution inference rule adds the new clause $C_1 \vee C_2$ to the set of clauses. Inspired by the theory of rewriting, if we restrict addition of facts to only those facts that make proofs of provable facts smaller, then $C_1 \vee C_2$ must be smaller than the two facts that were used to derive it. Restricting resolution inference in this way leads to the *ordered resolution* calculus.

Note that a set of facts is unsatisfiable if the empty clause, $\perp$, is provable. The ordering on facts is defined such that $\perp$ is the minimal element in the ordering. Since proof calculi inspired by rewriting are designed to generate all "small" facts, then if $\perp$ is provable, then it is explicitly generated. If it is not explicitly generated, then the set of generated (small) facts can be used to construct a model for the facts. This argument can be used to prove refutational completeness of ordered resolution. Several first-order proof calculi are designed, and proved refutationally complete, based on this same principle [5].

The theory of rewriting can be seen as a paradigm for saturating a set of facts with new facts, guided by an ordering on the universe of facts, such that certain minimal facts (such as $\perp$) are generated. This view is very helpful when developing heuristics for searching the space of (provable) facts for a particular one. As remarked above, the field of theorem proving is concerned with deriving $\perp$ to obtain a refutation. Now, as a second example, consider the problem of deciding if a conjunction of polynomial equality and inequality constraints is satisfiable in the theory of reals. First, consider the case when there are only linear equations and nonnegativity constraints on certain slack variables,

$$A \begin{bmatrix} \vec{x} \\ \vec{u} \end{bmatrix} = \vec{b}, \qquad \vec{u} > 0$$

where A is a $l \times (m+n)$ matrix, $\vec{x}$ is a $m \times 1$ vector, $\vec{u}$ is a $n \times 1$ vector, and $\vec{b}$ is a $l \times 1$ vector. We can prove that the above constraint is unsatisfiable over the reals if (and only if) we can find a linear expression $\vec{c}^T\vec{u}$ such that
(i) $\vec{c}^T\vec{u}$ can be written as a linear combination of the l linear expressions $A[\vec{x};\vec{u}]$, and
(ii) $\vec{c} \geq \vec{0}$ and $\vec{c} \neq \vec{0}$.
Such a linear expression can be thought of as the *witness* for unsatisfiability of the original set of constraints. If we find an ordering in which the witness, $\vec{c}^T\vec{u}$, is a minimal element in the set of all expressions that can be written as a linear combination of the l linear expressions $A[\vec{x};\vec{u}]$, then a rewriting-based saturation procedure will explicitly generate this linear expression. This is the idea behind the Simplex algorithm for linear constraints [15].

When the constraints are not necessarily linear, then there again exists a *witness* for unsatisfiability. This is stated by the Positivstellensatz [26, 31, 10]. Specifically, let P, Q, and R be sets of polynomials over $\mathbb{Q}[\vec{x}]$. The constraint

$$\{p = 0 : p \in P\} \cup \{q \geq 0 : q \in Q\} \cup \{r \neq 0 : r \in R\}$$

is unsatisfiable iff there exist polynomials p, q, and r such that $p + q + r^2 = 0$ where

$$\begin{array}{lll} p & \in & Ideal(P) \\ r & \in & [R] := \{\Pi_{i\in I}\, r_i : r_i \in R \text{ for all } i \in I\} \\ q & \in & Cone[Q] := \{\Sigma_{i\in I}\, p_i^2 q_i : q_i \in [Q], p_i \in \mathbb{Q}[\vec{x}] \text{ for all } i \in I\} \end{array}$$

If we find an ordering in which the witness, p, which is equal to $-q - r^2$, is a minimal element in the set of all elements in the ideal generated by P, then a rewriting-based saturation procedure will explicitly generate this witness. This is the idea behind the procedure for unsatisfiability checking based on Gröbner basis computation [34].

A crucial aspect in the above applications is the flexibility provided by the choice of ordering. The choice of ordering decides which facts are generated, and hence it determines if we will ever find a particular desired fact. This observation can be used to generate (equational) invariants of a continuous dynamical system. Consider the differential equations for circular motion: $\frac{dx}{dt} = y$, $\frac{dy}{dt} = -x$. A constant-of-motion, or an equational invariant, for this system would be a polynomial p over variables x, y such that $dp/dt = 0$. Let us order the above two equations backwards and write them as

$$y \to d(x) \qquad\qquad x \to -d(y)$$

We also have (infinite) rewrite rules that come from the definition of the derivative operator d. Consider the following two rules

$$yd(y) \to d(y^2)/2 \qquad\qquad xd(x) \to d(x^2)/2$$

Doing a critical-pair completion and computing a Gröbner basis for this system will yield an equation $d(x^2) + d(y^2) = 0$, which is the equational invariant for circular motion.

The rewriting philosophy is also used extensively in proving results about rewrite systems. Several decidable criterion have been recently developed for checking confluence of restricted classes of term rewriting systems [19, 21, 22, 20]. The characterizations of confluence are proved correct by showing that (a) if there is a counter-example to the characterization, then there will be a smaller counter-example and (b) all minimal counter-examples are explicitly checked.

4 Rewriting in Formal Modeling and Analysis

Rewriting provides both a language for formal modeling of systems, as well as a tool for simulating and analyzing the formal models.

The system Maude [12] is an example of a modeling and analysis tool based on rewriting logic. Maude is being extensively used to formally represent knowledge about biological processes. This knowledge is captured in the form of a Petrinet (represented in Maude). A Petrinet is a set of ground rewrite rules over a signature containing an associative and commutative (AC) symbol. A biochemical reaction is naturally a Petrinet transition [13, 23, 32]. Elaborate models of cell signaling pathways have been formalized in the Pathway Logic tool [32] that is built over Maude.

The rewrite-rule based models of biological processes pose several interesting analysis questions. Apart from questions about reachability, one is also interested in characterizing certain kinds of *steady state* behaviors. A steady state behavior is a rewrite derivation with certain properties. Flux balance analysis (FBA) is a commonly used technique for finding such steady state behaviors of Petrinets [29, 28]; see also [35] for an adaptation of FBA.

Biology also motivates a study of probabilistic rewrite systems. Again, a special case of (timed) stochastic Petrinets has been extensively studied [17, 18] and the same ideas can be possibly applied to stochastic extensions of general rewriting systems too.

The biology domain is an extremely rich source of challenging problems and extensions for the theory of rewriting. One such challenge is *learning* rewrite rules or models from available data on rewrite derivations. One could use it to learn models of disease propagation in humans and develop therapeutics based on the learned model.

5 Conclusion

The theory of rewriting is an important part of the foundation of computer science. It provides an important paradigm for algorithmic design and correctness and a uniform high-level view of several intricate algorithms and techniques. It also provides a useful modeling language, especially for the emerging discipline of Systems Biology [25, 27, 30].

References

1 W. W. Adams and P. Loustaunau. *An Introduction to Gröbner Bases*, volume 3 of *Graduate Studies in Mathematics*. American Mathematical Society, 1994.
2 L. Bachmair. *Canonical Equational Proofs*. Birkhäuser, Boston, 1991.
3 L. Bachmair and H. Ganzinger. Buchberger's algorithm: A constraint-based completion procedure. In *CCL*, volume 845 of *LNCS*. Springer, 1994.
4 L. Bachmair and H. Ganzinger. Rewrite-based equational theorem proving with selection and simplification. *J. of Logic and Computation*, 4:217–247, 1994.
5 L. Bachmair and H. Ganzinger. Resolution theorem proving. In *Handbook of Automated Reasoning*. Elsevier, 2001.
6 L. Bachmair, I.V. Ramakrishnan, A. Tiwari, and L. Vigneron. Congruence closure modulo AC. In *Proc. FroCoS*, volume 1794 of *LNAI*, pages 245–259. Springer, 2000.
7 L. Bachmair and A. Tiwari. D-bases for polynomial ideals over commutative noetherian rings. In *RTA*, volume 1103 of *LNCS*, pages 113–127. Springer, 1997.
8 L. Bachmair and A. Tiwari. Abstract congruence closure and specializations. In *Conf. on Automated Deduction, CADE 2000*, volume 1831 of *LNAI*, pages 64–78. Springer, 2000.
9 L. Bachmair, A. Tiwari, and L. Vigneron. Abstract congruence closure. *J. of Automated Reasoning*, 31(2):129–168, 2003.

10 J. Bochnak, M. Coste, and M.-F. Roy. *Real Algebraic Geometry*. Springer, 1998.
11 B. Buchberger. A critical-pair completion algorithm for finitely generated ideals in rings. In *Proc. Logic and Machines: Decision Problems and Complexity*, volume 171 of *LNCS*, pages 137–161, 1983.
12 M. Clavel et al. *Maude: Specification and Programming in Rewriting Logic*. http://maude.-csl.sri.com/manual/, SRI International, Menlo Park, CA, 1999.
13 V. Danos, J. Feret, W. Fontana, R. Harmer, and J. Krivine. Rule-based modelling of cellular signalling. In *Proc. CONCUR*, volume 4703 of *LNCS*, pages 17–41, 2007.
14 D. Detlefs, G. Nelson, and J. B. Saxe. Simplify: A theorem prover for program checking. *J. of the ACM*, 52(3):365–473, 2005.
15 B. Dutertre and L. de Moura. A fast linear-arithmetic solver for DPLL(T). In *CAV 2006*, volume 4144 of *LNCS*, pages 81–94, 2006.
16 C. Eder and J. Perry. Signature-based algorithms to compute Groebner bases. In *ISSAC*, 2011. arXiv:1101.3589v2.
17 D. T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J. Comp. Physics*, 22:403–434, 1976.
18 D. T. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *J. of Chemical Physics*, 115(4):1716–1733, 2001.
19 G. Godoy, R. Nieuwenhuis, and A. Tiwari. Classes of Term Rewrite Systems with Polynomial Confluence Problems. *ACM Trans. on Comp. Logic (TOCL)*, 5(2):321–331, 2004.
20 G. Godoy and A. Tiwari. Confluence of shallow right-linear rewrite systems. In *CSL 2005*, volume 3634 of *LNCS*, pages 541–556. Springer, 2005.
21 G. Godoy, A. Tiwari, and R. Verma. On the confluence of linear shallow term rewrite systems. In *STACS 2003*, volume 2607 of *LNCS*, pages 85–96. Springer, 2003.
22 G. Godoy, A. Tiwari, and R. Verma. Characterizing confluence by rewrite closure and right ground term rewrite systems. *AAECC*, 15(1):13–36, June 2004.
23 W. S. Hlavacek et al. Rules for modeling signal-transduction systems. *Sci STKE*, 344, 2006. PMID: 16849649.
24 D. Kapur. Shostak's congruence closure as completion. In *Rewriting Techniques and Applications, RTA 1997*, volume 1103 of *LNCS*, pages 23–37, 1997.
25 H. Kitano. Systems biology: A brief overview. *Science*, 295:1662–1664, 2002.
26 J. L. Krivine. Anneaux preordonnes. *J. Anal. Math.*, 12:307–326, 1964.
27 P. Lincoln and A. Tiwari. Symbolic systems biology: Hybrid modeling and analysis of biological networks. In *HSCC*, volume 2993 of *LNCS*, pages 660–672, 2004.
28 J.D. Orth, I. Thiele, and B.O. Palsson. What is flux balance analysis? *Nature Biotechnology*, 28:245–248, 2010.
29 B.O. Palsson. *Systems Biology: Properties of Reconstructed Networks*. Cambridge University Press, 2006.
30 C. Priami. Algorithmic systems biology. *CACM*, 52(5):80–88, 2009.
31 G. Stengle. A Nullstellensatz and a Positivstellensatz in semialgebraic geometry. *Math. Ann.*, 207, 1974.
32 C. L. Talcott. Pathway logic. In *Formal Meth. for Comp. Sys. Bio., SFM*, volume 5016 of *LNCS*, pages 21–53, 2008. `http://pl.csl.sri.com`.
33 A. Tiwari. *Decision procedures in automated deduction*. PhD thesis, State University of New York at Stony Brook, 2000.
34 A. Tiwari. An algebraic approach for the unsatisfiability of nonlinear constraints. In *CSL 2005*, volume 3634 of *LNCS*, pages 248–262. Springer, 2005.
35 A. Tiwari, C. Talcott, M. Knapp, P. Lincoln, and K. Laderoute. Analyzing pathways using SAT-based approaches. In *AB*, volume 4545 of *LNCS*. Springer, 2007.