Journal of Nonlinear Mathematical Physics Volume 13, Supplement (2006), 66–75 AGMF Tallin '05

Rewriting in Operads and PROPs

Lars HELLSTRÖM †

† *Sand 216, 881 91 Sollefteå, Sweden*
E-mail: lars.hellstrom@residenset.net

This article is part of the Proceedings of the Baltic-Nordic Workshop, **Algebra, Geometry and Mathematical Physics** *which was held in Tallinn, Estonia, during October 2005.*

Abstract

This paper is an informal collection of observations on how established rewriting techniques can be applied to or need to be adapted for use in non-associative algebras, operads, and PROPs.

Keywords: Rewriting, operad, diamond lemma, universal algebra, PROP, Lie algebra.

1 Introduction

Rewriting is the formal theory behind one of the most fundamental methods of mathematical calculation: to rewrite an expression in one or several steps, employing only a given set of established *rules* (identities), until a form for the expression acceptable as "an answer" is reached. While we do not often think of things that way, the activity as such is really something we spend a lot of time doing, although not always with the support of a formal theory. Typical assistance that one can get from that end is in proving that there always is some "answer", that this answer is unique, that a particular algorithm (or more practically: a particular route of calculations) always succeeds in finding an answer, etc. The theory also leads to methods for constructing effective models of algebraic structures, thus complementing the often very abstract definitions that textbooks have a tendency to give.

A striking feature of the subject is that the fundamental tools have a strong tendency to be reinvented, each time in a slightly different guise, which has produced a myriad of terminologies and formalisms. In commutative algebra, the main computational workhorse is the theory of Gröbner bases, whose utility springs from the fact that the Buchberger [4, 5] algorithm made it possible to effectively compute them. In more general rewriting (i.e., where it is called rewriting) that same thing is known as the Knuth–Bendix [8] completion procedure, and in associative algebra it is sometimes called the Composition Lemma of Bokut' [2] and other times called the Diamond Lemma of Bergman [1]. The common idea in all of the above is to consider the set of all minimal cases of expressions that can be rewritten in two different ways (the so-called critical pairs in Gröbner terminology, or equivalently 'ambiguities' in diamond lemma terminology), and examine whether both

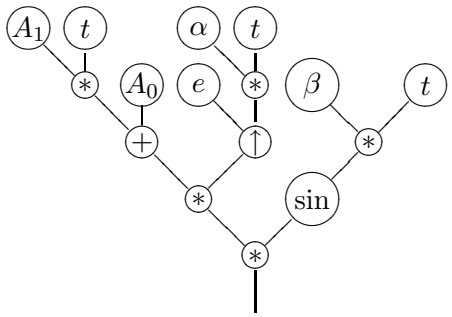


Figure 1. Term tree of $(A_1t + A_0)e^{\alpha t} \sin \beta t$

rewritten expressions simplify to the same thing. If they do not then we have discovered a nontrivial consequence of the known rules which may be added to the system as a new rule, and if they do then we know we have a complete system of rewriting rules, which leads to an algorithm for testing equivalence modulo these rules; either way one's knowledge of the situation increases. It is however known that in some noncommutative situations the procedure *cannot* terminate (a necessary condition for being an algorithm), so public attention has been somewhat focused on the "success story" of the Gröbner basis theory.

Here I will instead start in the associative realm and give a travelogue of the lands beyond it. As is often the experience of the mathematical explorer, these new lands turn out to extend primarily in an unexpected direction, and accordingly the non-associative rings that we have been aware of are not typical for the new geology. Instead much of it consist of operads, PROPs, and even more foreign materials, that (among other curious properties) can simultaneously be associative and non-associative. The good news is that the tools mentioned above generalise quite well to these new environments, so the surroundings are not as alien in nature as they may seem in substance.

2 Background

The classical model used for mathematical expressions derives very directly from the formalism of first order languages in mathematical logic, but should also be familiar to anyone who has studied computer languages or compiler construction. Here one usually speaks of *terms*, which come in three forms: (i) constants, (ii) variables, and (iii) a function applied to one or more terms. What in normal mathematical notation is the expression $(A_1t + A_0)e^{\alpha t} \sin \beta t$ should in the formal theory rather be transcribed as something like

$$*\Big(*\Big(+\big(*(A_1, t), A_0\big), \uparrow\big(e, *(\alpha, t)\big)\Big), \sin\big(*(\beta, t)\big)\Big),$$

where $+$ stands for the addition function $(a, b) \mapsto a + b$, $*$ for the multiplication function $(a, b) \mapsto ab$, and $\uparrow$ is the power function $(a, b) \mapsto a^b$. A more comprehensible (but unfortunately also more spacious) view of such things can be obtained as the *term tree* (or Abstract Syntax Tree, as it is called in Computer Science) of the expression. Figure 1 shows the tree form of the above term. Rules for term rewriting typically perform local transformations of such trees; for example applying distributivity to change $(A_1t + A_0)e^{\alpha t} \sin \beta t$ into

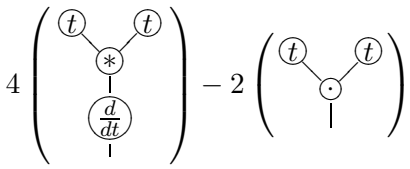


Figure 2. Formally multilinear tree

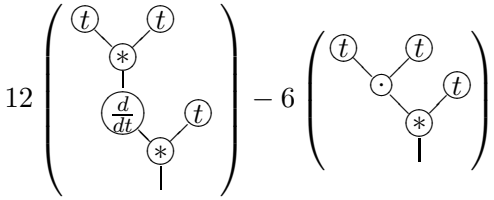


Figure 3. The result of $*$-ing the term in Figure 2 by $3t$ on the right

$(A_1te^{\alpha t} + A_0e^{\alpha t})\sin\beta t$ can be accomplished by locally changing the $+$ node and its parent $*$ node, to instead make the $+$ node the parent of two $*$ nodes. In the strict tree formalism, this also involves making a copy of the subtree rooted in the $\uparrow$ node, so that each of the two new $*$ nodes has its own copy, whereas in the DAG (Directed Acyclic Graph) formalism common subterms can be shared. Note that contemporary computer algebra systems generally do keep the above tree (or something very much like it) in memory as the representation of $(A_1t + A_0)e^{\alpha t}\sin\beta t$, so these matters have very practical applications.

There is however also another model—that of formally multilinear trees—that is important for many types of algebras. In this case only multilinear operations (multiplication, differentiation, etc.) occur as nodes in the tree. Expressions may also involve addition and multiplication by scalar, but these are treated as operations on a formal linear combination of trees, so that effectively an "expression" is a collection of trees with attached scalar coefficients. Figure 2 shows an example of such a formally multilinear tree with three different operations $*$, $\cdot$, $\frac{d}{dt}$, and also a constant t.

It may be remarked that all these formal terms can be regarded not only as elements of a set, but also as elements of an algebra, i.e., as elements of a set on which is defined certain operations. In the pure term model, these operations simply combine their operands into a larger tree whose new root carries the applied operation as decoration. While such operations may not seem to actually *do* anything, they from the standpoint of mathematical logic transport a problem from an abstract theoretical setting to the more concrete world of models. They also make it possible to speak of homomorphisms from formal terms to actual algebras, and thereby for example to interpret the set of all formal terms as a free object in (a supercategory of) the category of algebras being considered. In the formally multilinear trees model, the operations are less formal; addition and multiplication by scalar behaves just as in any free module. The remaining operations still combine trees to make a larger tree as in the pure model, but they also multiply the coefficients of all operands, and when an operand is a sum the result is (to ensure distributivity) the same as the sum of having applied the operation separately to each term of that operand; Figure 3 shows an example of this. While this multilinearity mostly reflects properties that operations are axiomatically required to have, and thus is a very good thing, it does

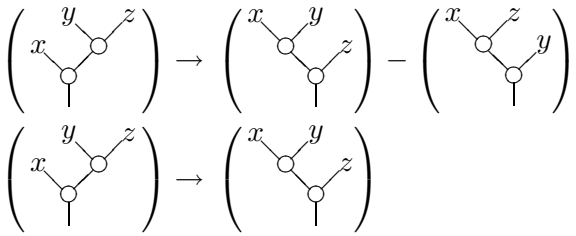


Figure 4. The Leibniz and associative identities as rewrite rules

have some unexpected consequences that will surface in Subsection 5.2 below.

3 An anticommutative conundrum

My first real encounter with rewriting theory came in the realm of associative algebra, with the diamond lemma formalism, so that is where I'm most at home (see e.g. [7]). Reputedly the associative setting is very hard, since there are unsolvable problems lurking around, but in practice I've always found it rather sensible. Concrete problems coming from applications typically can be solved, although it does require ingenuity as there is no general algorithmic solution. I have however at times found it strangely difficult to venture beyond the associative, and it was not until I got acquainted with operads that I understood why that was. As it turns out, this was due not to a property of the target, but to the nature of the origin—associative algebra. In order to view it from the outside, let us first make a little naive excursion into the non-associative.

One of the oldest and certainly in the literature most prolific classes of non-associative algebras are the Lie algebras. From a casual inspection it may seem as these should be quite similar, from a rewriting perspective, to an associative algebra with some commutation-like defining relation. Indeed, universal enveloping algebras of Lie algebras fit that description and are (via the proof of the Poincaré–Birkhoff–Witt theorem) known to be quite well behaved from a rewriting perspective, so one might expect lots of connections between the two realms. Admittedly the Jacobi identity $\big[[x,y],z\big] + \big[[z,x],y\big] + \big[[y,z],x\big] = 0$ is not very useful for rewriting (because it's too symmetric), but as an axiom of Lie algebras it can be replaced by the Leibniz identity $\big[x,[y,z]\big] = \big[[x,y],z\big] - \big[[x,z],y\big]$, and that fits in perfectly; in particular it plays the same role as the associativity identity $x(yz) = (xy)z$ in converting right-branching term trees to left-branching ones, see Figure 4. That far, it seems as though everything should work out the same.

What remains is the anticommutativity identity $[y,x] = -[x,y]$, which in this context can be classified as a commutativity-like identity, and indeed is one of the simpler in that bunch. When in the associative case one desires the anticommutativity in an exterior algebra generated by $\{\mathsf{a}_1,\dots,\mathsf{a}_n\}$, one merely introduces the $n(n+1)/2$ rewriting rules

$$\mathsf{a}_i^2 \to 0 \qquad \text{for all } i = 1,\dots,n, \tag{3.1a}$$

$$\mathsf{a}_i\mathsf{a}_j \to -\mathsf{a}_j\mathsf{a}_i \qquad \text{for all } j < i \tag{3.1b}$$

and that is sufficient; by employing these rules one arrives at a normal form for every expression. (Not an altogether trivial conclusion, but the calculations it takes to verify

it using the diamond lemma are quite straightforward.) Everything seems on track for a rewriting approach to Lie algebras.

It is quite possible to combine the anticommutativity rules (3.1) with a Leibniz rewrite rule

$$\big[x,[y,z]\big] \to \big[[x,y],z\big] - \big[[x,z],y\big] \tag{3.2}$$

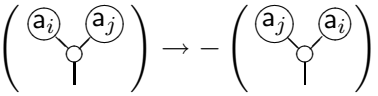
and study the corresponding rewrite theory, but the result is a disappointment, because it does not describe a Lie algebra; about the only things which anticommute are the generators $\mathsf{a}_1,\dots,\mathsf{a}_n$, whereas products of these are completely independent. What went wrong? Why do the anticommutativity relations not impose identities for longer products, as they do in an exterior algebra? For a long time I thought there was something special about the combination Leibniz identity and anticommutativity, because when one goes through the calculations of the Knuth–Bendix completion procedure—verifying that (3.1) and (3.2) form a complete set of rewrite rules—everything falls in place almost suspiciously well; it is as if they constitute a degenerate case in some very abstract variety of algebras. That conclusion, although mostly void of actual information, is very soothing: there's no need to worry about the case of Lie algebras, because it is anyway probably not representative of the generic behaviour. The conclusion is also completely wrong. It is not the Lie algebras that are special, but the associative ones!

One important detail that is different between the two arguments is the theorems that were used. For the exterior algebra (associative) I referred to the diamond lemma, whereas in the Lie algebra (not associative) I had to use Knuth–Bendix completion, because the diamond lemma requires associativity. If instead one tries the Knuth–Bendix completion on the exterior algebra system of rules, then something interesting happens. First, it turns out that the stated system of rules is *not* complete, but will have to be completed with rules

$$(x\mathsf{a}_i)\mathsf{a}_i \to 0 \qquad \text{for all } i=1,\dots,n, \tag{3.3a}$$

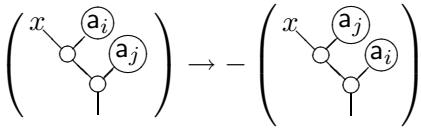
$$(x\mathsf{a}_i)\mathsf{a}_j \to -(x\mathsf{a}_j)\mathsf{a}_i \qquad \text{for all } j<i. \tag{3.3b}$$

Second, the calculations deriving these are not like the calculations for the diamond lemma, but the calculations involving only rules on the form (3.3) precisely parallels those from the diamond lemma, so this must be what was *really* being checked—not that rules like (3.1) suffice for defining an exterior algebra, but that rules like (3.3) do. Graphically the difference is between rules that on one hand say



(3.4)

and on the other



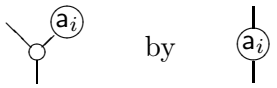
(3.5)

The reason the attempt at defining Lie algebras failed should now be clear; it is not anticommutativity like (3.4) that suffices for defining an exterior algebra, but rather the

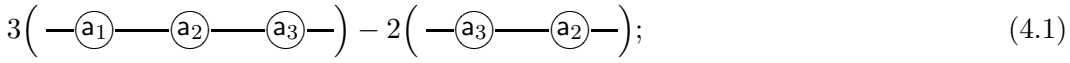
open-ended form (3.5), and this unfortunately does not have a counterpart for Lie algebras. Anticommutativity in exterior algebras is not merely a matter of exchanging leaves mounted on the same multiplication node, but also a matter of exchanging leaves between neighbouring multiplication nodes! Neither is anticommutativity in Lie algebras merely a matter of exchanging leaves mounted on the same multiplication node, but rather a matter of exchanging arbitrary operands (arbitrarily large trees) of a multiplication node. This accounts for a good deal of the differences, from a rewriting perspective, between the two types of algebra.

4 A different view of associativity

While the above has explained what became of the anticommutativity, it did not answer the question of why $\mathsf{a}_i\mathsf{a}_j \to -\mathsf{a}_j\mathsf{a}_i$ is treated as (3.5) by the diamond lemma, when the classical model of mathematical expressions rather say it should be interpreted as (3.4). The clue that made me understand what was happening came from exposure to the operad concept, because there one sometimes employs an alternative model, in which the generators of an algebra are *unary* operations and there really isn't any multiplication operation but only composition of operations. (In retrospect, this is not strange at all. Many important examples of associative algebras are operator algebras with composition as multiplication, and an operator is a unary operation.) The short step from (3.5) to this model is to replace each



which is mostly a notational transformation. Switching to a horizontal alignment of these operations, one finds that an expression such as $3\mathsf{a}_1\mathsf{a}_2\mathsf{a}_3 - 2\mathsf{a}_3\mathsf{a}_2$ is perhaps best thought of as the formal linear combination of paths (in the graph theory sense)



(4.1)

the isomorphism with the formal linear combination of words construction of the free associative algebra is now apparent.

Observation 1. *The ambiguities considered in the Diamond Lemma formalism are not consistent with the classical term model of associative algebra, but they are consistent with a model where each generator is a unary operation and 'multiplication' really is composition.*

Conclusion 1.1. *The natural setting for Bergman's diamond lemma is that of an algebra with unary operations.*

Having discovered this characterisation, an immediate question should be whether there is anything about the theorem which requires the operations to be unary. The answer to that turns out to be negative: the proof machinery as given in [7] has no problem with operations of arbitrary arity, one only has to generalise the basic objects from "term paths" (corresponding to monomials in the usual terminology) such as those in (4.1) to more general term trees.

Conclusion 1.2. *The diamond lemma can be generalised to support multilinear operations of arbitrary arity. The resulting theorem is about $\mathcal{R}$-linear operads.*

At this point, a difference between operad elements and classical terms needs to be pointed out: classical terms are syntactically complete expressions, but operad elements are more like operations or functions—you may have to plug some operand(s) in before you can get anything out. In the diagrams above this distinction has merely been hinted at by circling or not circling the labels at the ends of branches, but in the context of a generalised diamond lemma this distinction becomes important. Open-ended rules such as those in Figure 4 can be applied also in the interior of a term tree, whereas closed rules such as that in (3.4) only applies to entire branches.

In an operad, it is important to distinguish between elements of different arities, i.e., between elements that are "like" operations that take so-and-so many operands. Elements with arities greater than 1 compose to elements of even greater arities, so a lot of the things that happen do so at arities 3 and above. The Leibniz and associative identities are for example identities in arity 3 (even if composed from arity 2, i.e. binary, elements) and the resolution of their ambiguities is a calculation in arity 4.

Conclusion 1.3. *There is more of relevance to algebra and rewriting than what fits in arity zero.*

Despite the prudent appearance of this conclusion, it is somewhat heretical, because it contradicts the classical idea that "an algebra" consists of a carrier set A and some operations $f_i\colon A^{n_i} \longrightarrow A$, period. Elements of the carrier set can equivalently be viewed as constants or nullary operations, so from an operadic point of view this doctrine says that everything aims to prove things about arity zero elements, which is probably at best a curious idea. Associative algebra can be carried out entirely in arity 1, and lots of interesting things go on in higher arities, but it is true that some common algebraic questions are particularly questions about elements of a carrier set. These are often the ones that turn out too be awkward in an operad setting.

5 Difficulties and further extensions

Available space does not permit a similar treatment of PROPs, but as far as the actual rewriting is concerned the generalisation is straightforward: just replace trees as fundamental objects by more general graphs. The interpretation of these graphs as "expressions" is perhaps not immediately obvious, but straightforward enough once it has been explained, and it seems to compare rather well with the established notation systems (e.g. Sweedler notation) in this area. The difficulties are more in the algebra than in the rewriting.

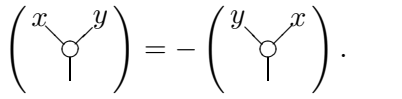
The remainder of this section is again primarily about operads and non-associative algebras, but PROPs resurface in the final paragraph.

5.1 Ordering

Observation 2. *There are no admissible total orderings of trees with arity 2 or higher.*

The need for admissible *total* orderings is most pronounced in the Gröbner basis formalism, where this ordering is used to single out a distinguished leading term from any

identity discovered during processing. Admissibility means primarily that the order relation is strictly compatible with multiplication, or as it becomes in the operadic setting: with composition. For elements of arity greater than 1 it also needs to include strict compatibility with permutation of inputs—if $\mu < \nu$ then $\mu \cdot \sigma < \nu \cdot \sigma$ for any permutation σ—but this is impossible for a total ordering to satisfy since if $\mu < \mu \cdot \sigma$ then the contradiction $\mu < \mu \cdot \sigma < \mu \cdot \sigma^2 < \cdots < \mu \cdot \sigma^n = \mu$, where n is the order of σ, follows. The diamond lemma avoids this predicament by allowing partial orders, but it cannot escape the fact that some identities are impossible to convert to rules in the traditional way. The immediate example of this is again Lie algebra anticommutativity, which is diagrammatically the following arity two identity:



(5.1)

If a rule $LHS \to RHS$ is made from this identity, then the result of any application of this rule is a term to which the rule can be applied again, so the rewriting process never terminates—is just gets stuck in a short cycle.

Conclusion 2.1. *The traditional definition that something is on normal form if no rewriting rule can be applied to it is insufficient for important classes of algebras. A new mechanism for designating a normal form is called for.*

One prototype for such a mechanism that may be possible to generically combine with the diamond lemma can be found in the theory of Gröbner–Shirshov bases (for a historical overview see e.g. [3]), where effectively the two operands are compared and only those terms where e.g. the larger is second count as being on normal form. Note however that this mechanism as used there only applies to arity zero elements. Whether it is trivial to extend to higher arities is at the time of writing unknown.

5.2 Non-linearity

Classically the algebraic structure of (possibly multivariate) functions under composition is called a *clone* (closed set of operations on, see e.g. [6, p. 127]); some well-known clones are: the set of polynomials over a given coefficient ring, the set of recursive number-theoretical functions, the set of primitive recursive number-theoretical functions, and the set of elementary functions (as in an elementary calculus textbook). Operads are more restricted than clones, in that they require that each operand is used exactly once in an expression, whereas clones permit arbitrary repetition. Thus p defined by $p(x) = x \cdot x$ is allowed in a clone as soon as there is multiplication, but not in an operad, since the x appears twice. One reason for this restriction is the desire to keep operads linear; squaring is not linear, but multiplication is linear in each operand separately, so it must be disallowed to repeat an operand. (While "clone" here appears to have a different etymological origin, it is striking that the same "no cloning" theorem holds in Multilinear Algebra as in Quantum Mechanics; the proof only needs linearity to go through.) The reason linearity is needed is that this brand of rewriting is based on the formally multilinear tree model; without it, there would be no way to compose with operands that are sums of

trees.[1]

One reason this aspect of the multilinear formalism needs to be brought up is that many identities used to define traditional classes of non-associative algebras are nonlinear, and this presents an obstruction to the application of operadic rewriting theory to them. One example is the class of *alternative algebras*, which have instead of associativity the identities

$$x^2y = x(xy) \qquad \text{and} \qquad (xy)y = xy^2,$$

that are required to hold for *all* elements x and y of the algebra. Every alternative algebra generalises to an operad, but there is no way to directly encode these defining identities as relations between elements in that operad, because neither side of either identity can be an operad element, due to the repetition of the variable x and y respectively. What to do?

In this particular case there is an operadic solution, because an alternative set of defining identities is

$$(x_1x_2)x_3 - x_1(x_2x_3) = (-1)^\sigma\big((x_{\sigma(1)}x_{\sigma(2)})x_{\sigma(3)} - x_{\sigma(1)}(x_{\sigma(2)}x_{\sigma(3)})\big) \qquad \text{for all permutations } \sigma \text{ of } \{1,2,3\}$$

that are multilinear and thus possible to encode as relations of arity 3 elements, but whether this is possible for all known classes of nonassociative algebras is an open question. It certainly seems as though the classical defining identities very often are nonlinear, and thus it might be necessary to develop new techniques for handling nonlinear identities. There are no counterparts in commutative algebra, but it is quite similar to the Polynomial Identities defining an associative PI-algebra, so that is probably where to look for inspiration.

A different approach for handling nonlinear identities is to introduce a formal "diagonal map" that clones its input, or at least does the closest approximation of a cloning that linearity permits. These diagonal maps, or *coproducts* as they are more commonly called, live in PROPs rather than in operads,[2] but the step to that generality is rather small and well worth the effort. One axiom system that has been treated this way is that of a group, where the axiom for the inverse $xx^{-1} = e = x^{-1}x$ requires repeating the operand x, and the result is rather interesting: one arrives at the axiom system for Hopf algebras! (Alternatively one may arrive at cocommutative Hopf algebras, or some other variation; the choice of axioms for the coproduct is not unique.) These formal diagonal map axiomatisations tend to be weaker than the original concept, but may preserve many of the interesting properties; a Hopf algebra is in this sense a weakened form of the group algebra of a free group. Whether such bialgebra variants of e.g. alternative or Jordan algebras are of any interest remains to be seen.

[1] Well, one can in principle do some sort of generalised formally multilinear tree model in which each operation comes with rules describing the equivalent of distributivity for that particular operation, but that quickly gets a lot more complicated, and one has to stop somewhere. It appears p-Lie-algebras would require something along those lines, though.

[2] Yes, there are operads corresponding to coalgebras, so you can house coproducts in operads, but I'd rather say these are co-operads. Anyway we'll also want a product, and *then* a full-blown PROP is unavoidable.

Acknowledgements

Most of the research underlying this travelogue was carried out when the author was a postdoc at the Mittag-Leffler institute, during the Noncommutative Geometry (NOG) programme, partially funded by the ESF. Some of the rest has been done while visiting the Department of Mathematics at Umeå University and the Centre for Mathematical Sciences at Lund University respectively. The author's participation at the AGMF workshop was supported for by the Department of Mathematics at Umeå University. Partial support of the Swedish Foundation for International Cooperation in Research and Higher Education (STINT) and the Crafoord Foundation is also gratefully acknowledged.

References

[1] Bergman G M, The Diamond Lemma for Ring Theory, *Adv. Math.* **29** (1978), 178–218.

[2] Bokut' L A, Embeddings into simple associative algebras (Russian), *Algebra i Logika* **15**, no. 2 (1976), pp. 117–142 and 245. English translation in *Algebra and Logic*, pp. 73–90.

[3] Bokut' L A and Kolesnikov P S, Gröbner–Shirshov bases: from inception to the present time. (Russian. English, Russian summary) *Zap. Nauchn. Sem. S.-Peterburg. Otdel. Mat. Inst. Steklov. (POMI)* **272** (2000), *Vopr. Teor. Predst. Algebr i Grupp. 7*, 26–67, 345; translation in *J. Math. Sci. (N. Y.)* **116** (2003), no. 1, 2894–2916.

[4] Buchberger B, *Ein Algorithmus zum Auffinden der Basiselemente der Restklassenringes nach einem nulldimensionalen Polynomideal* (German: An Algorithm for Finding a Basis for the Residue Class Ring of a Zero-Dimensional Polynomial Ideal), Doctoral Dissertation, University of Innsbruck, Institute for Mathematics, 1965.

[5] Buchberger B, Introduction to Gröbner Bases, in *Gröbner bases and applications*, London Math. Soc. Lecture Note Ser., **251**, Cambridge Univ. Press, 1998, 3–31.

[6] Cohn P M, *Universal Algebra*, Harper & Row, Publishers, New York–London, 1965.

[7] Hellström L, *The Diamond Lemma for Power Series Algebras* (doctorate thesis), 2002; ISBN 91-7305-327-9; HTTP://`abel.math.umu.se/~lars/diamond/thesis.pdf` or ditto `/thesis.ps.gz`.

[8] Knuth D E and Bendix P B, Simple word problems in universal algebras, in *Computational Problems in Abstract Algebra (Proc. Conf., Oxford, 1967)*, Editor: Leech J, Pergamon, Oxford, 1970, 263–297. Also in *Automation of Reasoning Vol. 2*, Editors: Siekmann J H and Wrightson G, Springer, 1983, 342–376; ISBN 3-540-12044-0.