General Polynomial Reduction with THEOREMA Functors

Applications to Integro-Differential Operators and Polynomials

Bruno Buchberger*

Georg Regensburger†

Markus Rosenkranz†

Loredana Tec*

Abstract. **We outline a prototype implementation of the algorithms for integro-differential operators and polynomials presented in [10], programmed in the functor language of the TH∃OREM∀ system [5].**

General Polynomial Reduction

We use a fixed Gröbner basis for normalizing integro-differential operators. Gröbner bases were invented by Buchberger [2, 3] for commutative polynomials and reinvented in [1] for noncommutative ones. Among the systems implementing noncommutative Gröbner bases [6], none of them allows two features that are important for our present setting:

- Polynomials in infinitely many variables
- Reduction modulo infinite systems

Our generic approach encompasses commutative/noncommutative polynomials as well as one/two-sided reduction. Polynomial algebras are formulated as monoid algebras over a field K and a monoid W via the functor MonoidAlgebra[K, W], leading to:

1. Commutative polynomials
   W = additive monoid $\mathbb{N}^n$.
2. Noncommutative polynomials
   W = word monoid $\{x_1,\dots,x_n\}^*$.
3. Exponential polynomials
   W = additive monoid $\mathbb{N}\times\mathbb{C}$.

Sample computations:

1. Commutative bivariate case:
   $(2x+y)*(2x-y)=4x^2-y^2$

```
TS_In[25]:= Compute[⟨⟨2, ⟨1, 0⟩⟩, ⟨1, ⟨0, 1⟩⟩⟩ *_ℙ
              ⟨⟨2, ⟨1, 0⟩⟩, ⟨-1, ⟨0, 1⟩⟩⟩]
TS_Out[25]= ⟨⟨4, ⟨2, 0⟩⟩, ⟨-1, ⟨0, 2⟩⟩⟩
```

2. Noncommutative bivariate case:
   $(2x+y)*(2x-y)=4x^2-2xy+2yx-y^2$

```
TS_In[32]:= Compute[⟨⟨2, ⟨"x"⟩⟩, ⟨1, ⟨"y"⟩⟩⟩ *_ℙ
              ⟨⟨2, ⟨"x"⟩⟩, ⟨-1, ⟨"y"⟩⟩⟩]
TS_Out[32]= ⟨⟨4, ⟨x, x⟩⟩, ⟨-2, ⟨x, y⟩⟩, ⟨2, ⟨y, x⟩⟩, ⟨-1, ⟨y, y⟩⟩⟩
```

3. Exponential polynomials:
   $(2xe^{\sqrt{2}x})*(4x^3e^{-\sqrt{2}x})=8x^4$

```
TS_In[74]:= Compute[⟨⟨2, ⟨1, √2⟩⟩⟩ *_ℙ ⟨⟨4, ⟨3, -√2⟩⟩⟩]
TS_Out[74]= ⟨⟨8, ⟨4, 0⟩⟩⟩
```

Polynomial reduction is realized by a noncommutative adaption of reduction rings (rings with so-called reduction multipliers in the sense of [4, 11]; for a noncommutative approach along different lines, we refer to [7].

```
red_G[f, g] (* the reduction of f modulo g *) =
  f -_P lrdm_P[f, g] *_P g *_P rrdm_P[f, g]
```

Example:

Let $f(x,y)=x^3y^2+5$, $g(x,y)=x^2y+xy$.

- Commutative case: $rdm(f,g)=xy$,
  $red(f,g)=-x^2y^2+5$
- Noncommutative case: $lrdm(f,g)=x$,
  $rrdm(f,g)=y, red(f,g)=-xxyy+5$

The TH∃OREM∀ System

The generic implementation of monoid algebras with reduction multipliers is realized through functors whose principle and implementation in the TH∃OREM∀ version of higher order predicate logic were introduced by Buchberger. The TH∃OREM∀ system is designed as an integrated environment for doing mathematics [5], in particular

- proving,
- computing,
- solving

in various domains of mathematics. Its core language is higher-order predicate logic, containing a natural programming language such that algorithms can be coded and verified in a unified formal frame. In this logic-internal programming language, functors are a powerful tool for realizing a modular and generic build-up of hierarchical domains in mathematics. A functor is viewed as a function that produces a new domain from given domains by defining operations in the new domain in terms of operations in the underlying domains.

The following functor takes a linearly ordered alphabet L as input domain and builds the corresponding words over it (here $\bar{\xi}$, $\bar{\eta}$ are sequence variables, i.e. they can be instantiated with finite sequences of terms). The new domain W has the following properties:

```
Definition["Word Monoid", any[L],
 LexWords[L] = Functor[W, any[v, w, ξ, η, ξ̄, η̄],
    s = ⟨⟩
    ─────────────────────────────────────
    ∈_W[w] ⟺ ⋀{ is-tuple[w]
                 ∀_{i=1,…,|w|} ∈_L[w_i]
    ⬚_W = ⟨⟩
    v *_W w = v ≍ w
    (⟨η, η̄⟩ >_W ⟨⟩) ⇔ True
    (⟨⟩ >_W ⟨η̄⟩) ⇔ False
    (⟨η, η̄⟩ >_W ⟨ξ, ξ̄⟩) ⇔ ⋁{ η >_L ξ
                              (η = ξ) ⋀ ⟨η̄⟩ >_W ⟨ξ̄⟩
                                                        ]]
```

- $W[\in]$: all letters are in L
- $W[\square]$: neutral element
- $W[*]$: concatenation
- $W[>]$: lexicographic ordering

The Monoid Algebra is the crucial functor that builds up polynomials. After adding reduction multipliers, the operations for handling Gröbner bases are added by virtue of an extension functor (a functor that leaves previous operations unchanged and adds new ones).

```
Definition["Monoid Algebra", any[K, W],
 MonoidAlgebra[K, W] = Functor[P, any[c, d, f, g, …],
    s = ⟨⟩
    ─────────────────────────────────────
    ∈_P[f] ⟺ ⋀{ is-tuple[f]
                 ∀_{i=1,…,|f|} ⋀{ is-tuple[f_i]
                                   |f_i| = 2
                                   ∈_K[(f_i)_1]
                                   ∈_W[(f_i)_2]
                                   (f_i)_1 ≠ 0_K
                 ∀_{i=1,…,|f|-1} (f_i)_2 >_W (f_{i+1})_2
    1_P = ⟨⟨1_K, ⬚_W⟩⟩                                        ]]
    0_P = ⟨⟩
    ⟨⟩ *_P g = ⟨⟩
    f *_P ⟨⟩ = ⟨⟩
    ⟨⟨c, ξ⟩, m̄⟩ *_P ⟨⟨d, η⟩, n̄⟩ =
      (⟨⟨c *_K d, ξ *_W η⟩⟩ +_P ⟨c, ξ⟩ *_P ⟨n̄⟩) +_P ⟨m̄⟩ *_P ⟨⟨d, η⟩, n̄⟩
    f +_P g = …
```

Integro-Differential Operators

The notion of integro-differential operators was introduced in [10] as a generalization of the "Green's polynomials" of [9]. They are particularly useful for treating boundary problem for LODEs as they express both the problems statement (differential equation and boundary conditions) and its solution operator (an integral operator usually called "Green's operator"). The integro-differential operators are realized by a suitable quotient of noncommutative polynomials over a given integro-differential algebra.

```
Definition["FreeIntDiffOp", any[ℱ, K],
 FreeIntDiffOp[ℱ, K] = where[ℒ = DegWords["∂" ↦
      ("∫" ↦ DoubleBasis[Basis[ℱ], CharBasis[K]])],
   ℳ = MonoidAlgebra[K, ℒ],
   Functor[𝒜, any[b, c, f, w̄, …],
    s = ⟨⟩
    ─────────────────────────────────────
    ∈_𝒜[f] ⟺ ∈_ℳ[f]
    …
    ─────────────────────────────────────
    ⟨⟩ ⋄_𝒜 f (* action of int-diff operator *) = f
    ⟨w̄, "∂"⟩ ⋄_𝒜 f = ⟨w̄⟩ ⋄_𝒜 ∂_ℱ f                        ]]]
    ⟨w̄, "∫"⟩ ⋄_𝒜 f = ⟨w̄⟩ ⋄_𝒜 ∫_ℱ f
    ⟨w̄, ⌊c⌋⟩ ⋄_𝒜 f = eval_ℱ[f, c] ·_ℱ ⟨w̄⟩ ⋄_𝒜 1_ℱ
    ⟨w̄, ⌈b⌉⟩ ⋄_𝒜 f = ⟨w̄⟩ ⋄_𝒜 (ew_𝒜[b] *_ℱ f)
```

```
Definition["IntDiffOp", any[ℱ, K],
 IntDiffOp[ℱ, K] = where[𝒜 = FreeIntDiffOp[ℱ, K],
   QuotAlg[𝒜, 𝒢_𝔾𝕣]]]
```

An ordinary integro-differential algebra $(\mathcal{F},\partial,\int)$ is a differential algebra with a K-linear operation $\int\colon\mathcal{F}\to\mathcal{F}$, such that

$$\partial\int f=f \quad\text{and}\quad (\int f)(\int g)=\int f\int g+\int g\int f.$$

In order to build up the integro-differential operators, we first consider the monoid algebra for the word monoid over the infinite alphabet consisting of the letters ∂ and $\int$ along with all basis elements $x^ne^{\lambda x}$ $(n\in\mathbb{N},\lambda\in\mathbb{C})$ of the exponential polynomials and all multiplicative functionals φ. Then we factor out the nine (parametrized) rewrite rules:

$fg\to f\cdot g$	$\partial f\to\partial\cdot f+f\partial$
$\varphi\psi\to\psi$	$\partial\varphi\to 0$
$\varphi f\to(\varphi\cdot f)\varphi$	$\partial\int\to 1$
$\int f\int\to(\int\cdot f)\int-\int(\int\cdot f)$	
$\int f\partial\to f-\int(\partial\cdot f)-(\mathrm{E}\cdot f)\,\mathrm{E}$	
$\int f\varphi\to(\int\cdot f)\varphi$	

These rules form a Gröbner basis in the underlying polynomial ring. Computing in the quotient algebra is realized by using the corresponding normal forms.

Integro-Differential Polynomials

The integro-differential polynomials over an algebra [8] form a commutative algebra. They model nonlinear differential and integral operators with an indeterminate u, so a typical element would be $\int(x^4uu'^2\int(xe^{3x}u^2u'^3\int u))$. One can describe extentions of an integro-differential algebra by forming suitable quotients of the integro-differential polynomials. We are currently working on an implementation based on the functors presented here.

Acknowledgements

This work was supported by the Austrian Science Fund FWF under the SFB grants F1302 and F1322.

We would also like to thank Manuel Kauers for assembling the practical LaTeX class file used for preparing this poster.

References

[1] G. M. Bergman. The diamond lemma for ring theory. *Adv. in Math.*, 29(2):179–218, 1978.

[2] B. Buchberger. An Algorithm for Finding the Bases Elements of the Residue Class Ring Modulo a Zero Dimensional Polynomial Ideal. PhD thesis, Univ. of Innsbruck, 1965. English translation in *J. Symbolic Comput.*, 41(3-4):475–511, 2006.

[3] B. Buchberger. Ein algorithmisches Kriterium für die Lösbarkeit eines algebraischen Gleichungssystems. *Aequationes Math.*, 4:374–383, 1970. English translation in B. Buchberger, F. Winkler (eds.), Gröbner Bases and Applications, Cambridge University Press, 1998.

[4] B. Buchberger. Groebner rings and modules. In S. Maruster, B. Buchberger, V. Negru, and T. Jebelean, eds., *Proc. of SYNASC 2001*, 2001.

[5] B. Buchberger et al. Theorema: Towards computer-aided mathematical theory exploration. *J. Appl. Log.*, 4(4):359–652, 2006.

[6] V. Levandovsky. Gröbner basis implementations. Functionality check and comparison. *http://www.ricam.oeaw.ac.at/Groebner-Bases-Implementations*, 2008.

[7] K. Madlener and B. Reinert. Non-commutative reduction rings. *Rev. Col. Mat.*, 33(1):27–49, 1999.

[8] M. Rosenkranz and G. Regensburger. Solving and factoring boundary problems for linear ordinary differential equations in differential algebras. *J. Symbolic Comput.*, 43(8):515–544, 2008.

[9] M. Rosenkranz. A new symbolic method for solving linear two-point boundary value problems on the level of operators. *J. Symbolic Comput.*, 39(2):171–199, 2005.

[10] M. Rosenkranz and G. Regensburger. Integro-differential polynomials and operators. In D. Jeffrey, ed., *Proc. of ISSAC'08*. ACM Press, 2008.

[11] S. Stifter. Gröbner bases of modules over reduction rings. *J. Algebra*, 159(1):54–63, 1993.

A Fourth Order Boundary Problem

Given $f\in C^\infty[0,1]$, we want to find the unique $u\in C^\infty[0,1]$ such that

$$\begin{cases} u'''' = f,\\ u(0)=u''(0)=u(1)=u''(1)=0.\end{cases}$$

The operator $G\colon f\mapsto u$, known as the Green's operator of the problem, can be computed [8] by normalizing the polynomial $(1-P)\int\int\int\int$, with

$$P=\tfrac{1}{6}x^3\lfloor 1\rfloor\partial\partial-\tfrac{1}{6}x^3\lfloor 0\rfloor\partial\partial+\tfrac{1}{2}x^2\lfloor 0\rfloor\partial\partial-\tfrac{1}{6}x\lfloor 1\rfloor\partial\partial-\tfrac{1}{3}x\lfloor 0\rfloor\partial\partial+x\lfloor 1\rfloor-x\lfloor 0\rfloor+\lfloor 0\rfloor,$$

where $\lfloor 0\rfloor$, $\lfloor 1\rfloor$ denote evaluation at 0 and 1, respectively.

Thus, we obtain

$$G=\tfrac{1}{6}x^3\lfloor 1\rfloor\int x+\tfrac{1}{6}x\lfloor 1\rfloor\int x^3-\tfrac{1}{2}x\lfloor 1\rfloor\int x^2+\tfrac{1}{3}x\lfloor 1\rfloor\int x-\tfrac{1}{6}x^3\lfloor 1\rfloor\int-\tfrac{1}{2}x^2\int x+\tfrac{1}{2}x\int x^2-\tfrac{1}{6}\int x^3+\tfrac{1}{6}x^3\int$$

by our implementation.

* RISC, Johannes Kepler Universität · † RICAM, Austrian Academy of Sciences · Altenberger Straße 69 · Linz, Austria