The homalg project

Mohamed Barakat, Markus Lange-Hegermann
(TU Kaiserslautern, RWTH-Aachen)

barakat@mathematik.uni-kl.de
markus.lange.hegermann@rwth-aachen.de

What is the homalg project?

The homalg project is a multi-author open source[1] software project for constructive homological algebra. Mainly written in GAP4 it allows the use of external programs and other computer algebra systems (CASs) for specific time critical tasks. Although the central part of the source code is the formalization of abstract notions like ABELian categories, our focus lies on concrete applications ranging from linear control theory to commutative algebra and algebraic geometry.

The stable packages of the project are distributed with GAP [GAP12] and the newer packages can be downloaded from the project homepage [hpa12].

Current applications

The first goal of the project was the construction of spectral sequences of filtered complexes (e.g. GROTHENDIECK spectral sequences) and the filtration they induce on total (co)homologies. Of cource, such a construction consists of many subalgorithms.

The core package homalg contains algorithms for (co)homology, connecting homomorphisms, long exact (co)homology sequences, filtrations, spectral sequences, etc., which are implemented abstractly, independent of the specific ABELian category. Other algorithms like derived functors (Ext, Tor, ...), CARTAN-EILENBERG resolutions, hyper derived functors, YONEDA products, etc., are implemented for categories having enough projectives or injectives. Extension packages implement the ABELian categories of finitely presented (f.p.) modules (e.g. D-modules), of f.p. multigraded modules, and (in a yet preliminary form) of coherent sheaves on toric varieties. These packages provide further algorithms to compute context specific properties and invariants like deciding torsion-freeness, reflexivity, locally freeness, freeness (QUILLEN-SUSLIN), purity, the computation of projective dimension, degree, purity filtration, etc. We use these properties in applications to algebraic system theory, an ongoing work with QUADRAT and CLUZEAU. Building on this infrastructure the project supports applications to algebraic geometry. This includes a constructive version of the BGG-correspondence, TATE resolutions of coherent sheaves on projective schemes, their characteristic classes and cohomology, higher direct images under morphisms to affine spaces, divisors, etc. Recently, GUTSCHE contributed a package for toric varieties.

Ideas behind the homalg project

The meta idea

Mathematical software should comply to mathematical ideas, regardless of how abstract they are.

An example

We exemplify this statement by the following piece of code taken from the homalg project

```
RightDerivedCofunctor(
 Functor_Hom_for_fp_modules, "Ext" );
```

Before we could write such a code we taught the following notions to the computer. For the second line we need the notion of the category of finitely presented modules over a computable commutative ring with its (internal) Hom-functor. The first line needs the more abstract notions ABELian categories with enough projectives, additive functors, and their right derived functors (regardless of the specific ABELian category). These two lines of code install several methods to compute Ext as a functor applicable to f.p. modules, their maps, and even complexes (of complexes) thereof.

The basic mathematical idea

The basic notion is that of a **computable ABELian category**, i.e., an ABELian category for which all disjunctions and existential quantifiers appearing in the axioms of an ABELian category are algorithmic. Given that, all

[1] All current packages are licensed under GPL-2.

constructions which only depend on the category being ABELian become computable. See [Bar09] for a construction of spectral sequences of filtered complexes based on the notion of generalized morphisms. This replaces diagram chasing of elements, as objects of general ABELian categories are not a priori sets[2].

Basic examples of computable ABELian categories are categories of finitely presented modules over a ring R equipped with an algorithm to solve linear systems $B = XA$ with coefficients in R. Examples for such rings are fields (GAUSSian algorithm), EUCLIDean rings (HNF), and commutative and noncommutative rings admitting a GRÖBNER basis algorithm [BLH11, BR08].

The challenge

This means that these abstract homological algorithms gradually specialize to matrix and GRÖBNER basis algorithms. But would it then not be more efficient to directly hardcode such matrix algorithms? This question is in fact a particular instance of what ABBOTT and BIGATTI described in [AB12, p. 11] as

> the challenge of reconciling the traditionally conflicting goals of (mathematical) abstraction and efficiency.

Knowledge based computation

We address this challenge by what we call "knowledge based computation", which we will now explain.

Our goal is to write an extremely simple and mathematically readable code which gets **optimized during runtime**. This is contrary to the conventional paradigm that an efficient program is a result of an optimized code[3] or an optimization during compile time.

Special input often causes special unforeseeable setups to occur during runtime. By runtime optimization we mean the use of mathematical knowledge applicable in special setups to shortcut further unnecessary computations. For this to work we did two things. First, we organized mathematical objects according to the mathematical hierarchy and realized them as **learning objects** which can accumulate knowledge during their lifetime. Second, we started teaching the system mathematical knowledge (lemmas, propositions, theorems) at all levels of the mathematical hierarchy[4] in form of **logical methods**. Each such method is installed globally allowing it to influence many different algorithms.

During runtime algorithms generate knowledge for the involved objects. And as soon as such knowledge is created even more knowledge is immediately deduced by the logical methods known to the system. Furthermore, mathematically related objects are connected as they are created inside the system; this allows the instantaneous propagation of knowledge between related objects, even if they lie in different abstraction layers.

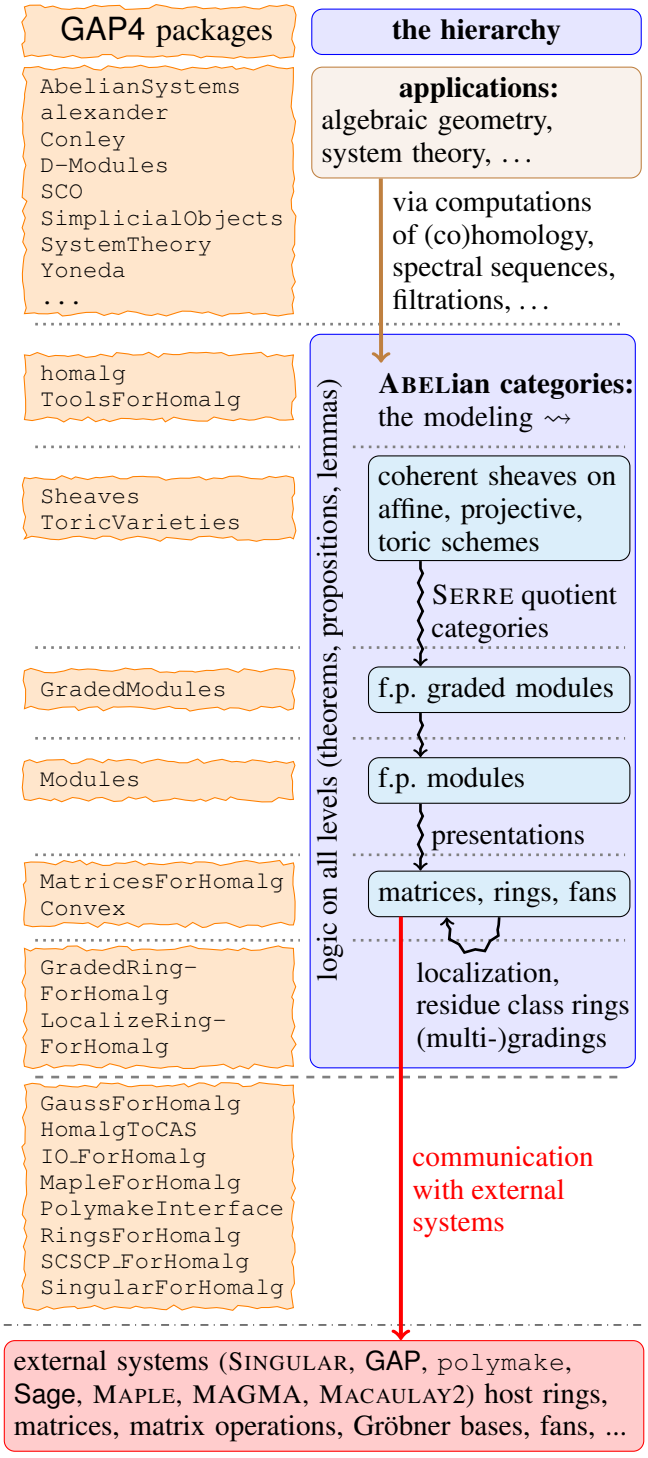

Figure 1: *The* `homalg` *project*

To maximize the benefit of mathematical knowledge we extensively make use of lazy evaluation[5]. This technology often avoids unnecessary not yet triggered computations, as it allows objects to await knowledge. There we make use of the fact that, contrary to the most general case, many computations (e.g., of GRÖBNER bases) become obsolete in particular cases, very often only identifiable during runtime.

We explain this by an example. For $R = k[x_1, \ldots, x_n]$ with a computable field k there are algo-

[2] Note that embedding theorems into module categories are in general not constructive.

[3] Quoting DONALD KNUTH: "... premature optimization is the root of all evil."

[4] ring/module elements, rings, matrices, (graded) modules, coherent sheaves, morphisms, complexes, spectral sequences, functors, ...

[5] The converse is eager evaluation, a common practice in computer algebra.

rithms composed of expensive subalgorithms to compute $\mathrm{Ext}^i_R(M,N)$ for finitely generated R-modules M, N. However, the system can combine knowledge to deduce that certain compositions of algorithms become unnecessary. For example, the following three theorems predict $\mathrm{Ext}^i_{k[x,y]}(\mathrm{Hom}_{k[x,y]}(M, k[x,y]), N)$ for $i > 0$.

(1) The global dimension of $k[x_1, \ldots, x_n]$ is n.

(2) If R is of global dimension 2 then $\mathrm{Hom}_R(M, R)$ is projective.

(3) A module H is projective iff $\mathrm{Ext}^i_R(H, N) = 0$ for all R-modules N and $i > 0$.

Hence, for $i > 0$

$$\mathrm{Ext}^i_{k[x,y]}(\overbrace{\mathrm{Hom}_{\underbrace{k[x,y]}_{\mathrm{gl.dim}=2 \text{ by (1)}}}(M, k[x,y])}^{\text{projective by (2)}}, N) \underbrace{= 0}_{\text{by (3)}}.$$

A logical deduction system has of course its computational cost, but it is negligible when it often succeeds in avoiding GRÖBNER basis computations which are, at least in the worst case, far more expensive. Furthermore, the cost of logical deduction grows much slower than the cost of GRÖBNER bases as a function on the input size.

We sum up this subsection by our slogan:

> Do not teach the computer the matrix algorithm. But teach it enough mathematical knowledge and let it figure out the most efficient matrix algorithm you know by itself. And eventually, the system will surprise you.

Intrinsic modeling

Another key idea of the project is to realize higher mathematical objects in a "coordinate free" way. For example, a module does not store only one set of generators but all generating sets which got created during its lifetime, together with enough information to create their transition matrices, once needed. If the internal presentation of an object is switched[6] all internal presentations of related objects[7] are automatically (but lazily) readapted. This is a constructive way of encoding mathematical objects *intrinsically*. It is not only extremely useful in certain applications but also necessary for the development of complex dynamical algorithms.

Why GAP4?

For the following two reasons we chose GAP4 as the high level language [BL98] for the `homalg` project.

GAP4 supports a remarkable object oriented programming paradigm which we like to describe as a **mathematical object orientedness**. Contrary to standard object oriented languages (like C++ or python) methods are not installed in the objects but are attached to so-called operations. The idea is that objects in GAP4 specialize their type during their lifetime. This allows the method selector to apply special algorithms which are faster than more generic ones.

The second decisive feature of the GAP4 language are the so-called immediate methods. We like describing them as a second level of execution in which the normal, usually expensive methods are halted and these presumably "zero-cost" methods are triggered. These are exactly the methods we use to infer and propagate knowledge until no further knowledge is deducible.

For example, to encode that projective modules over local domains are free we currently add the entry

```
[ [ IsProjective ],
  HomalgRing,
  [ [ IsLocal, IsIntegralDomain ] ],
  "imply", IsFree ]
```

to the list `LogicalImplicationsForModules`.

The presence of both features, which is crucial to the `homalg` project, are unique to the GAP4-language. The lack of similar functionality forced us to abandon a previous version of `homalg` in MAPLE. It also prevented us from choosing Sage, even though interfaces to external systems would have been available.

Software design = mathematical hierarchy

The software design of the project complies to the mathematical hierarchy. The basic notion of a computable ABELian category is implemented abstractly in the core package `homalg`. An implementation of a concrete ABELian category consists of two parts. The first part provides all algorithms necessary for its computability as an ABELian category. The second part implements special algorithms and logical methods only applicable to this specific category. Different concrete categories are implemented in different packages (`Modules`, `GradedModules`, `Sheaves`, ...). They all share the abstract algorithms implemented in `homalg`. In this sense, mathematical definitions dictate the interface between data structures in our implementation. Using these interfaces, the categories of coherent sheaves are modeled as SERRE quotient categories of the categories of graded modules, which are modeled on the categories of modules, and these are finally modeled on the categories of presentation matrices [BLH11].

For encoding these high level mathematical structures we mainly need bookkeeping objects for knowledge storage. Hence we do not rely on GAP4 for time critical issues, except for the external communication and for the propagation of knowledge, two processes which are highly optimized in GAP4.

We succeeded in separating the technical part of the project from the mathematical content. The packages and their dependencies reflect the mathematical hierarchy (cf. Figure 1). The modular design of the project has the convenient side effect of making the big amount of code easily extendable and maintainable.

[6] typically to a more economic one, speeding up subsequent computations.

[7] here, e.g., module maps, with the affected module being their source or target.

Delegation and multiple computer algebra systems
Another advantage of the modular design of the project is the ability to outsource rings, matrices with all their operations (sums, products, solving linear systems, ...) to external dedicated CASs like SINGULAR [DGPS12], MACAULAY2 [GS], MAGMA [BCP97], or MAPLE [Wat00]. The combinatorial computations for toric varieties are outsourced to `polymake` [GJ00].

For performance reasons it is important to minimize the communication between GAP and these external systems. Therefore, computed matrices are not sent back to GAP (except for user output). Instead, GAP refers to external data through pointers, and only integers (e.g., sizes of matrices) or boolean values (e.g., whether a matrix is zero or not) are sent back to GAP. This minimal information is sufficient for GAP to steer the algorithms and is enough for the logic to work.

A further abstraction layer in the project allows different ways of physical connections to external systems. Currently the standard way is to use IO-streams supported by NEUNHÖFFER's `IO`-package. Recently more robust C-level interfaces were written by our colleagues BÄCHLER, GUTSCHE, HORN, LÜBECK, NEUNHÖFFER, and SCHÖNEMANN to connect to MAPLE, `polymake`, and SINGULAR.

Future

The next step is to implement SERRE quotient categories following [BLH] and use them to model categories of coherent sheaves on varieties with a COX ring (toric varieties, MORI dream spaces, ...). Using this setup we want to implement categories of coherent sheaves equivariant under (discrete) group actions (utilizing the strength of GAP).

We plan to formalize further parts of homological algebra, e.g., to make the use of derived categories and dg-categories[8] constructive and to exploit constructive versions of derived equivalences in a more systematic way (the BGG-correspondence was a particular case). For this we need data structures for tilting objects and the support for rings that occur as their endomorphism rings. A first application will be the construction of BEILINSON monads and the BEILINSON spectral sequences (and the induced filtrations).

Dreaming further we like to imagine things like étale cohomology, the derived RIEMANN-HILBERT correspondence, perverse sheaves, GROTHENDIECK's six operations, ... becoming constructive.

References

[AB12] John Abbott and Anna M. Bigatti, *New Flavours of* CoCoA, *Computeralgebra-Rundbrief*, 50:9–12, March 2012.

[Bar09] Mohamed Barakat, *Spectral filtrations via generalized morphisms*, submitted (`arXiv:0904.0240`), 2009.

[BCP97] Wieb Bosma, John J. Cannon, and Catherine Playoust, *The Magma algebra system. I. The user language*, JSC **24** (1997), no. 3–4, 235–266, Computational algebra and number theory (London, 1993). MR 1 484 478

[BL98] Thomas Breuer and Steve Linton, *The GAP4 type system: organising algebraic algorithms*, ISSAC '98: Proceedings of the 1998 international symposium on Symbolic and algebraic computation (New York, NY, USA), ACM, 1998, pp. 38–45.

[BLH] Mohamed Barakat and Markus Lange-Hegermann, *Gabriel morphisms and the computability of Serre quotients with applications to coherent sheaves*, (in preparation).

[BLH11] ———, *An axiomatic setup for algorithmic homological algebra and an alternative approach to localization*, J. Algebra Appl. **10** (2011), no. 2, 269–293, (`arXiv:1003.1943`).

[BR08] Mohamed Barakat and Daniel Robertz, `homalg` – *A meta-package for homological algebra*, J. Algebra Appl. **7** (2008), no. 3, 299–317, (`arXiv:math.AC/0701146`). MR 2431811 (2009f:16010)

[DGPS12] W. Decker, G.-M. Greuel, G. Pfister, and H. Schönemann, SINGULAR *3-1-5 — A computer algebra system for polynomial computations*, (`http://www.singular.uni-kl.de`), 2012.

[GAP12] The GAP Group, *GAP – Groups, Algorithms, and Programming, Version 4.5.6*, 2012, (`http://www.gap-system.org`).

[GJ00] Ewgenij Gawrilow and Michael Joswig, *polymake: a framework for analyzing convex polytopes*, Polytopes—combinatorics and computation (Oberwolfach, 1997), DMV Sem., vol. 29, Birkhäuser, Basel, 2000, (`http://www.polymake.org`), pp. 43–73. MR 1785292 (2001f:52033)

[GS] Daniel R. Grayson and Michael E. Stillman, *Macaulay2, a software system for research in algebraic geometry*, (Available at `http://www.math.uiuc.edu/Macaulay2/`).

[hpa12] The homalg project authors, *The* `homalg` *project*, (`http://homalg.math.rwth-aachen.de/`), 2003–2012.

[Wat00] Waterloo Maple Inc., *Maple*, 1996, 1998, 2000, 57 Erb Street West, Waterloo, ON N2L 6C2, Canada.

[8] differentially enriched categories