A Computer Algebra System : Risa/Asir *

Masayuki Noro

1 An overview of Risa/Asir

Risa/Asir is a tool for performing various computations in mathematics and engineering applications. The development of Risa/Asir started in 1989 at FUJITSU and now the source code is also available at no cost. Currently, Kobe distribution is the most active branch in the development of Risa/Asir. Risa/Asir is characterized as follows:

1. An environment for large scale and efficient polynomial computation.

   Risa/Asir consists of the Risa engine for performing operations on mathematical objects and an interpreter for programs written in the Asir user language. In Risa/Asir, polynomials are represented in two different internal forms: the recursive representation and the distributed representation. Polynomial factorization and GCD are based on the former representation, and computations related to the Gröbner basis are based on the latter representation. Ground fields of polynomial rings can be composed of The field of rationals, algebraic number fields and finite fields are available as ground fields of polynomial rings.

2. A platform for parallel and distributed computation.

   In order to combine mathematical software systems, we previously proposed the OpenXM (Open Message eXchange for Mathematics) protocol. Risa/Asir acts as both an OpenXM client and an OpenXM server. The Risa/Asir OpenXM client API provides a way to call functions in external OpenXM servers. Using multiple Risa/Asir OpenXM servers, one can perform parallel and distributed computation in an attempt to achieve linear speedup. Conversely, other OpenXM clients can call functions in the Risa/Asir server.

3. Open source software.

   The source code of Risa/Asir is completely open, and algorithms and implementations can be verified if necessary. The addition of new codes is simple.

Risa/Asir runs on various platforms such as Linux, FreeBSD, Sun and Windows. The whole source code is available from `http://www.math.kobe-u.ac.jp/OpenXM`. The on-line help is available from the Risa/Asir command line or from the menu bar on Windows. Manuals are located at `http://www.math.kobe-u.ac.jp/OpenXM/Current/doc/index-doc.html`.

*This article is a summary of two papers[1][2].

2 Functionality of Risa/Asir

Among various functions on algebraic computation, we explain the implementations of two main functions: polynomial factorization and Gröbner basis computation over various ground fields.

2.1 Operations over fields of characteristic 0

Polynomial arithmetics, polynomial GCD and factorization are implemented over our own implementation of integer and rational number arithmetics. In multivariate factorization, we implemented a simplified version of Wang method to estimate the leading coefficient of a factor to avoid the leading coefficient problem.

Risa/Asir provides a univariate factorization over algebraic number fields represented by a successive extension. As an application, the splitting field computation of a univariate polynomial is implemented. The impelented algorithm is an improvement of the algorithm proposed by Trager. In the present algorithm, we utilize non square-free norms to obtain partial (not necessarily irreducible) factorization. The algorithm finally requires univariate factorization of a polynomial over the rationals, however this factorization is often difficult because the polynomial contains many spurious factors. At present, Risa/Asir is not optimized to factor polynomials of this type. In this case, a factorization in the PARI library implementing the knapsack factorization algorithm can be used, and the splitting field can be computed efficiently even in such cases [1].

2.2 Computation over finite fields

In Risa/Asir finite fields are represented in several ways.

1. $GF(p)$ (p: a prime $p < 2^{29}$)
2. $GF(p)$ (p: an arbitrary prime)
3. $GF(2^n)$ (n: small)
4. $GF(p^n)$ (p: an arbitrary prime, n: small integer)
5. $GF(p^s)$ (p: a small prime, $p^s < 2^{16}$)

The type 1 is used internally in various modular algorithms. The type 2, 3 and 4 are general purpose representations. Multiplication in $GF(2^n)$ is implemented using Karatsuba algorithm extensively. The type 5 has been introduced for efficient implementation of multivariate factorization over finite fields of small characteristic. When we attempt to factor a polynomial over such a field, we often have insufficient evaluation points. In this case we have to extend the ground field. If the ground field and its extension are both represented by a type 5 representation, we can expect that field arithmetics are performed efficiently in both fields. We presented a new polynomial time algorithm to factor bivariate polynomials over a finite field is presented. A combination of the Berlekamp-Hensel method and the new algorithm under a type 5 representation enables us to efficiently factor a certain class of polynomials, including hard-to-factor polynomials.

[1] A special configure option is required for linking the new version of PARI library.

2.3 Gröbner basis computation

We have incorporated various algorithms and improvements for Gröbner basis computation. For the Buchberger algorithm, we have implemented well-known criteria to detect unnecessary pairs, the sugar strategy, trace algorithms by modular computation, stabilization of strategy by combining homogenization and the trace algorithm, and efficient content reduction during normal form computation. We also have an experimental implementation of F_4 algorithm. Furthermore, we implemented several types of change of ordering algorithms. Among these, `tolex()` implements a modular FGLM algorithm, which avoids intermediate coefficient swells during the ordinary FGLM algorithm and realizes efficient computation of lexicographic Gröbner bases for zero-dimensional ideals.

By supplying an option `Demand` with a directory name, generated basis elements are placed in the directory. Although this requires additional cost in order to read the basis elements required for normal form computations, the total amount of memory is smaller than the case in which all basis elements are placed in memory, which may enable a very large computation, generating numerous intermediate basis elements.

As an application of Gröbner basis computation and polynomial factorization, we implemented primary ideal decomposition and prime decomposition of the radical of an ideal. These functions can be used to decompose solutions of systems of algebraic equations into irreducible components.

We implemented fundamental arithmetics, the Buchberger algorithm and the minimal polynomial computation over Weyl algebra, the ring of differential operators having polynomial coefficients. Using these arithmetics we implemented an efficient algorithm for computing the b-function of a polynomial.

2.4 Parallel and distributed computation via OpenXM

Risa/Asir provides OpenXM API for executing funtions on other mathematical software. In OpenXM, mathematical software systems are wrapped as server stack machines. As an OpenXM client, Risa/Asir dispatches a request to a server and receives the result. Inputs and outputs are represented as CMO (Common Mathematical Object format) objects. The set of stack machine commands also contains various control operations, such as server invocation, termination, interruption and restarting. Usually, each mathematical software system has its own user language. OpenXM provides stack machine commands for executing a command string and receiving a result as a human readable character string, thus wrapping a mathematical software system as an OpenXM server is relatively easy. OpenXM protocol is completely open, and any program can implement the protocol. OpenXM specifies a procedure for robust interruption and restarting of execution, which enables clients to be reset safely from any state.

Communication between an OpenXM server and a client can be realized by various methods: files, TCP/IP, MPI, PVM, RPC and linking a subroutine library. The Risa/Asir subroutine library `libasir.a` contains functions simulating the stack machine commands supported in `ox_asir`, the OpenXM asir server.

2.5 Integration of Risa/Asir and other OpenXM servers

Asir OpenXM contrib (asir-contrib) is a collection of wrappers of functions in external OpenXM servers. Using asir-contrib, an external function can be called from Asir with-

out knowing that the function is located in an external server. Currently, the following functions (including Asir built-in functions) are provided in OpenXM:

- Operations on Integers

  `idiv`,`irem` (division with remainder), `ishift` (bit shifting), `iand`,`ior`,`ixor` (logical operations), `igcd`,(GCD by various methods such as Euclid's algorithm and the accelerated GCD algorithm), `fac` (factorial), `inv` (inverse modulo an integer), `random` (random number generator by the Mersenne twister algorithm).

- Ground Fields

  Arithmetics on various fields: the rationals, $\mathbf{Q}(\alpha_1, \alpha_2, \ldots, \alpha_n)$ (α_i is algebraic over $\mathbf{Q}(\alpha_1, \ldots, \alpha_{\mathtt{i}-1})$), $GF(p)$ (p is a prime of arbitrary size), $GF(2^n)$.

- Operations on Polynomials

  `sdiv` , `srem` (division with remainder), `ptozp` (removal of the integer content), `diff` (differentiation), `gcd` (GCD over the rationals), `res` (resultant), `subst` (substitution), `umul` (fast multiplication of dense univariate polynomials by a hybrid method with Karatsuba and FFT+Chinese remainder), `urembymul_precomp` (fast dense univariate polynomial division with remainder by the fast multiplication and the precomputed inverse of a divisor),

- Polynomial Factorization `fctr` (factorization over the rationals), `modfctr`, `fctr_ff` (univariate factorization over finite fields), `af` (univariate factorization over algebraic number fields), `sp` (splitting field computation).

- Groebner basis

  `dp_gr_main` , `nd_gr_trace` (Groebner basis computation of a polynomial ideal over the rationals by the trace lifting), `dp_gr_mod_main` , `nd_gr` (Groebner basis over small finite fields), `tolex` (Modular change of ordering for a zero-dimensional ideal), `tolex_gsl` (Modular rational univariate representation for a zero-dimensional ideal), `dp_f4_main`, `dp_f4_mod_main`, `nd_f4` (F_4 over the rationals and small finite fields).

- Ideal Decomposition

  `primedec`, `primedec_mod` (Prime decomposition of the radical), `primadec` (Primary decomposition of ideals by Shimoyama/Yokoyama algorithm).

- Quantifier Elimination

  `qe` (real quantifier elimination in a linear and quadratic first-order formula), `simpl` (heuristic simplification of a first-order formula).

- Visualization of curves

  `plot` (plotting of a univariate function), `ifplot` (plotting zeros of a bivariate polynomial), `conplot` (contour plotting of a bivariate polynomial function).

- Miscellaneous functions

  `det` (determinant), `qsort` (sorting of an array by the quick sort algorithm), `eval`, `deval` (evaluation of a formula containing transcendental functions such as `sin`, `cos`, `tan`, `exp`, `log`) `pari(roots)` (finding all roots of a univariate polynomial), `pari(lll)` (computation of an LLL-reduced basis of a lattice).

- D-modules (D is the Weyl algebra)

  `sm1.gb` (Gröbner basis), `sm1.syz` (syzygy), `sm1.bfunction`,`bfunction` (the global b-function of a polynomial) `sm1.restriction` in the derived category of D-modules, `sm1.slope`, `sm1.sm1(annfs)` (Annhilating ideal of f^s), `sm1.sm1(schreyer)` (free resolution by the Schreyer method), `sm1.sm1(characteristic)` (Characteristic variety), `sm1.sm1(integration)` in the derived category, `sm1.sm1(res-dual)` (Dual as a D-module).

- Cohomology groups

  `deRham` (The de Rham cohomology groups of $\mathbf{C}^n \setminus V(f)$, `ext` (Ext modules for a holonomic D-module M and the ring of formal power series).

- Differential equations

  Helping to derive and prove `combinatorial` and special function identities, `sm1.gkz` (GKZ hypergeometric differential equations), `sm1.appell1`, `sm1.appell4` (Appell's hypergeometric differential equations), `sm1.generalized_bfunction` (indicial equations), `sm1.rank` (Holonomic rank), `sm1.rrank` (Holonomic rank of regular holonomic systems), `dsolv_dual`, `dsolv_starting_terms` (series solutions of holonomic systems).

- OpenMATH support

  `om_xml` (CMO to OpenMATH XML), `om_xml_to_cmo` (OpenMATH XML to CMO).

- Homotopy Method

  `phc.phc` (Solving systems of algebraic equations by numerical and polyhedral homotopy methods).

- Toric ideal

  `tigers.tigers` (Enumerate all Gröbner basis of a toric ideal. Finding test sets for integer program),

- Communications

  `ox_launch` (starting a server), `ox_launch_nox`, `ox_shutdown`, `ox_launch_generic`, `generate_port`, `try_bind_listen`, `try_connect`, `try_accept`, `register_server`, `ox_rpc`, `ox_cmo_rpc`, `ox_execute_string`, `ox_reset` (reset the server), `ox_intr`, `register_handler`, `ox_push_cmo`, `ox_push_local`, `ox_pop_cmo`, `ox_pop_local`, `ox_push_cmd`, `ox_sync`, `ox_get`, `ox_pops`, `ox_select`, `ox_flush`, `ox_get_serverinfo`

References

[1] M. Noro, A Computer Algebra System Risa/Asir. Algebra, Geometry and Software, M. Joswig and N. Takayama (eds.), Springer, 147-162 (2002).

[2] M. Maekawa, M. Noro, N. Takayama and Y. Tamura The Design and Implementation of OpenXM-RFC 100 and 101. Computer Mathematics (Proc. ASCM2001), World Scientific, 102-111 (2001).