decomp.lib 0.98

A Singular libary for
Functional Decomposition of Polynomials

Christian Gorzel, University of Münster, 2010

1 Library decomp.lib

Purpose: Functional Decomposition of Polynomials

Author: Christian Gorzel, University of Muenster
email: gorzelc@math.uni-muenster.de

Overview: This library implements functional uni-multivariate decomposition of multivariate polynomials.

A (multivariate) polynomial f is a composite if it can be written as $g \circ h$ where g is univariate and h is multivariate, where $\deg(g), \deg(h) > 1$.

Uniqueness for monic polynomials is up to linear coordinate change $g \circ h = g(x/c - d) \circ c(h(x) + d)$.

If f is a composite, then `decompose(f);` returns an ideal (g,h); such that $\deg(g) < \deg(f)$ is maximal, $(\deg(h) \geq 2)$. The polynomial h is, by the maximality of $\deg(g)$, not a composite.

The polynomial g is univariate in the (first) variable vvar of f, such that deg_vvar(f) is maximal.

`decompose(f,1);` computes a full decomposition, i.e. if f is a composite, then an ideal $(g_1, \ldots, g_m, h)$ is returned, where g_i are univariate and each entry is primitive such that $f = g_1 \circ \ldots \circ g_m \circ h$.

If f is not a composite, for instance if $\deg(f)$ is prime, then `decompose(f);` returns f.

The command `decompose` is the inverse: `compose(decompose(f,1))==f`.

Recall, that Chebyshev polynomials of the first kind commute by composition.

The decomposition algorithms work in the tame case, that is if char(basering)=0 or p:=char(basering) > 0 but deg(g) is not divisible by p. Additionally, it works for monic polynomials over Z and in some cases for monic polyomials over coefficient rings.
See `is_composite` for examples. (It also works over the reals but there it seems not be numerical stable.)

More information on the univariate resp. multivariate case.

Univariate decomposition is created, with the additional assumption $\deg(g), \deg(h) > 1$.

A multivariate polynomial f is a composite, if f can be written as $g \circ h$, where g is a univariate polynomial and h is multivariate. Note, that unlike in the univariate case, the polynomial h may be of degree 1.
E.g. $f = (x + y)^2 + 2(x + y) + 1$ is the composite of $g = x^2 + 2x + 1$ and $h = x + y$.

If `nvars(basering)>1`, then, by default, a single-variable multivariate polynomial is not considered to be the same as in the one-variable polynomial ring; it

will always be decomposed. That is:

```
> ring r1=0,x,dp;
> decompose(x3+2x+1);
x3+2x+1
```

but:

```
> ring r2=0,(x,y),dp;
> decompose(x3+2x+1);
_[1]=x3+2x+1
_[2]=x
```

In particular:
`is_composite(x3+2x+1)==1;` in `ring r1` but
`is_composite(x3+2x+1)==0;` in `ring r2`.

This is justified by interpreting the polynomial decomposition as an affine Stein factorization of the mapping $f : k^n \to k, n \geq 2$.

The behaviour can changed by the some global variables.

`int DECMEHTH;` choose von zur Gathen's or Kozen-Landau's method.
`int MINS;` compute f = g o h, such that h(0) = 0.
`int IMPROVE;` simplify the coefficients of g and h if f is not monic.
`int DEGONE;` single-variable multivariate are considered uni-variate.

See `decompopts;` for more information. Additional information is displayed if `printlevel > 0`.

Procedures:

decompopts	displays resp. resets global options
decompose	[complete] functional decomposition of poly f
is_composite	predicate, is f a composite polynomial?
chebyshev	the nth Chebyshev polynomial of the first kind
compose	compose entries of ideal

Auxiliary procedures:

makedistinguished	transforms f to a var-distinguished polynomial
divisors	intvec [increasing] of the divisors d of n
randomintvec	random intvec size n, [non-zero] entries in {a,b}
findtransformation	transforms f to a var-distinguished polynomial
maxdegs	maximal degree for each variable of the poly f

References:

D. Kozen, S. Landau: Polynomial Decomposition Algorithms,
J. Symb. Comp. (1989), 7, 445-456.

J. von zu Gathen: Functional Decomposition of Polynomials: the Tame Case,
J. Symb. Comp. (1990), 9, 281-299.

J. von zur Gathen, J. Gerhard: Modern computer algebra,
Cambridge University Press, Cambridge, 2003.

1.1 decompopts

Usage: decompopts(); or decompopts("reset");

Return: nothing

Note: in the first case, it shows the setting of the control parameters;
in the second case, it kills the user-defined control parameters and
resets to the default setting which will then be diplayed.

int DECMETH; Method for computing the univariate decomposition
0 : (default) Kozen-Landau
1 : von zur Gathen

int IMPROVE Choice of coefficients for the decomposition
$(g_1, \ldots, g_l, h)$ of a non-monic polynomials f.
0 : leadcoef(g_1) = leadcoef(f) and $g_2, \ldots, g_l, h$ are monic
1 : (default), content(g_i) = 1

int MINS
$f = g \circ h, (g_1, \ldots, g_m, h)$ of a non-monic polynomials f.
0 : g(0) = f(0), h(0) = 0 [ueberlegen fuer complete]
1 : (default), g(0)=0, h(0) = f(0)
2 : Tschirnhaus

int DECORD; The order in which the decomposition will be computed
0 : minfirst
1 : (default) maxfirst

int DEGONE; decompose also polynomials built on linear ones
0 : (default)
1 :

Example:

```
LIB "decomp.lib";
decompopts();
↦
↦   === Global variables for decomp.lib ===
↦
↦  -- DECORD (int) not defined, implicitly 1
↦  -- MINS (int) not defined, implicitly 0
↦  -- IMPROVE (int) not defined, implicitly 1
```

1.2 decompose

Usage: decompose(f); f poly
decompose(f,1); f poly

Return: poly, the input, if f is not a composite
ideal, if the input is a composite

Note: computes a full decomposition if called by the second variant

See: compose

Example:

```
LIB "decomp.lib";
ring r2 = 0,(x,y),dp;
decompose(((x3+2y)^6+x3+2y)^4);
↦ _[1]=x24+4x19+6x14+4x9+x4
↦ _[2]=x3+2y
// complete decomposition
decompose(((x3+2y)^6+x3+2y)^4,1);
↦ _[1]=x2
↦ _[2]=x2
↦ _[3]=x6+x
↦ _[4]=x3+2y
// ------------------------------------------------------------------------
// decompose over the integers
ring rZ = integer,x,dp;
↦ // ** You are using coefficient rings which are not fields.
↦ // ** Please note that only limited functionality is available
↦ // ** for these coefficients.
↦ // **
↦ // ** The following commands are meant to work:
↦ // ** - basic polynomial arithmetic
↦ // ** - std
↦ // ** - syz
↦ // ** - lift
↦ // ** - reduce
decompose(compose(ideal(x3,x2+2x,x3+2)),1);
↦ _[1]=x3
↦ _[2]=x2-1
↦ _[3]=x3+3
// ------------------------------------------------------------------------
// prime characteristic
ring r7 = 7,x,dp;
decompose(compose(ideal(x2+x,x7)));   // tame case
↦ _[1]=x2+x
↦ _[2]=x7
// ------------------------------------------------------------------------
decompose(compose(ideal(x7+x,x2)));   // wild case
↦ x14+x2
// ------------------------------------------------------------------------
ring ry = (0,y),x,dp;     // y is now a parameter
compose(x2+yx+5,x5-2yx3+x);
↦ x10+(-4y)*x8+(4y2+2)*x6+(y)*x5+(-4y)*x4+(-2y2)*x3+x2+(y)*x+5
decompose(_);
↦ _[1]=1/4*x2+(-1/4y2+5)
↦ _[2]=2*x5+(-4y)*x3+2*x+(y)
// Usage of variable IMPROVE
ideal J = x2+10x, 64x7-112x5+56x3-7x, 4x3-3x;
decompose(compose(J),1);
↦ _[1]=x2+10*x
↦ _[2]=64*x7-112*x5+56*x3-7*x
↦ _[3]=4*x3-3*x
int IMPROVE=0;
exportto(Decomp,IMPROVE);
decompose(compose(J),1);
↦ _[1]=1099511627776*x2+10485760*x
```

```
↦ _[2]=x7-7/64*x5+7/2048*x3-7/262144*x
↦ _[3]=x3-3/4*x
```

1.3 is_composite

Usage: is_composite(f); f poly

Return: int
1, if f is decomposable
0, if f is not decomposable
-1, if char(basering)>0 and deg(f) is divisible by char(basering) but no decomposition has been found.

Note: The last case means that it could exist a decomposition f=g o h with char(basering)|deg(g), but this wild case cannot be decided by the algorithm. Some additional information will be displayed when called by the user.

Example:

```
LIB "decomp.lib";
ring r0 = 0,x,dp;
is_composite(x4+5x2+6);    // biquadratic polynomial
↦ 1
is_composite(2x2+x+1);     // prime degree
↦  The degree is prime.
↦ 0
// ------------------------------------------------------------------------
// polynomial ring with several variables
ring R = 0,(x,y),dp;
// ------------------------------------------------------------------------
// single-variable multivariate polynomials
is_composite(2x+1);
↦  The polynomial is linear
↦ 0
is_composite(2x2+x+1);
↦ 1
// ------------------------------------------------------------------------
// prime characteristic
ring r7 = 7,x,dp;
is_composite(compose(ideal(x2+x,x14)));     // is_composite(x14+x7);
↦ 1
is_composite(compose(ideal(x14+x,x2)));     // is_composite(x14+x2);
↦ // -- Warning: wild case, cannot decide whether the polynomial has a
↦ // -- decomposition goh with deg(g) divisible by char(basering) = 7.
↦ -1
```

1.4 chebyshev

Usage: chebyshev(n); n int, n >= 0
chebyshev(n,c); n int, n >= 0, c number, c!=0

Return: poly, the [monic] nth Chebyshev polynomial of the first kind.
The polynomials are defined in the first variable, say x, of the basering.

Note: The (generalized) Chebyshev polynomials of the first kind can be defined by the recursion: $C_0 = c,\ C_1 = x,\ C_n = 2/c \cdot x \cdot C_{n-1} - C_{n-2},\ n \geq 2, c \neq 0$.

These polynomials commute by composition: $C_m \circ C_n = C_n \circ C_m$.
For c=1, we obtain the standard (non monic) Chebyshev polynomials T_n which satisfy $T_n(x) = \cos(n \cdot \arccos(x))$.
For c=2 (default), we obtain the monic Chebyshev polynomials P_n which satisfy the relation $P_n(x + 1/x) = x^n + 1/x^n$.
By default the monic Chebyshev polynomials are returned: P_n =`chebyshev(n)` and T_n =`chebyshev(n,1)`.
It holds $P_n(x) = 2 \cdot T_n(x/2)$ and more generally $C_n(c \cdot x) = c \cdot T_n(x)$
That is `subst(chebyshev(n,c),var(1),c*var(1))= c*chebyshev(n,1)`.

If `char(basering) = 2`, then $C_0 = 1, C_1 = x, C_2 = 1, C_3 = x$, and so on.

Example:

```
LIB "decomp.lib";
ring r = 0,x,lp;
// The monic Chebyshev polynomials
chebyshev(0);
↦ 2
chebyshev(1);
↦ x
chebyshev(2);
↦ x2-2
chebyshev(3);
↦ x3-3x
// These polynomials commute
compose(chebyshev(2),chebyshev(6)) ==
compose(chebyshev(6),chebyshev(2));
↦ 1
// The standard Chebyshev polynomials
chebyshev(0,1);
↦ 1
chebyshev(1,1);
↦ x
chebyshev(2,1);
↦ 2x2-1
chebyshev(3,1);
↦ 4x3-3x
// ------------------------------------------------------------------------
// The relation for the various Chebyshev polynomials
5*chebyshev(3,1)==subst(chebyshev(3,5),x,5x);
↦ 1
// ------------------------------------------------------------------------
// char 2 case
ring r2 = 2,x,dp;
chebyshev(2);
↦ 1
chebyshev(3);
↦ x
```

1.5 compose

Usage: compose(I); I ideal,
compose(f1,...,fn); f1,...,fn poly

Assume: the ideal consists of n=ncols(I) >= 1 entries,
where I[1],...,I[n-1] are univariate in the same variable
but I[n] may be multivariate.

Return: poly, the composition I[1](I[2](...I[n]))

Note: this procedure is the inverse of decompose

See: decompose

Example:

```
LIB "decomp.lib";
ring r = 0,(x,y),dp;
compose(x3+1,x2,y3+x);
↦ y18+6xy15+15x2y12+20x3y9+15x4y6+6x5y3+x6+1
// or the input as one ideal
compose(ideal(x3+1,x2,x3+y));
↦ x18+6x15y+15x12y2+20x9y3+15x6y4+6x3y5+y6+1
```

1.6 makedistinguished

Usage: makedistinguished(f,vvar); f, vvar poly; where vvar is a ring variable

Return: (poly, ideal): the transformed polynomial and an ideal defining the map which reverses the transformation.

Purpose: let vvar = var(1). Then f is transformed by a random linear coordinate change phi = (var(1), var(2)+c_2*vvar,...,var(n)+c_n*vvar)
such that phi(f) = f o phi becomes distinguished with respect to vvar. That is, the new polynomial contains the monomial vvar^d, where d is the degree of f. If already f is distinguished w.r.t. vvar, then f is left unchanged and the re-transformation is the identity.

Note 1: (this proc correctly works independent of the term ordering.) to apply the reverse transformation, either define a map or use substitute (to be loaded from poly.lib).

Note 2: If p=char(basering) > 0, then there exist polynomials of degree d>=p, e.g. $(p-1)x^py + xy^p$, that cannot be transformed to a vvar-distinguished polynomial. In this case, *p random trials will be made and the proc may leave with an ERROR message.

Example:

```
LIB "decomp.lib";
int randval = system("--random");  // store initial value
system("--random",0815);
ring r = 0,(x,y),dp;
poly g;
map phi;
// ------------------------------------------------------------------------
// Example 1:
poly f = 3xy4 + 2xy2 + x5y3 + x + y6;    // degree 8
// make the polynomial y-distinguished
g, phi = makedistinguished(f,y);
g;
↦ x5y3+5x4y4+10x3y5+10x2y6+5xy7+y8+y6+3xy4+3y5+2xy2+2y3+x+y
```

```
phi;
↦ phi[1]=x-y
↦ phi[2]=y
// to reverse the transformation apply the map
f == phi(g);
↦ 1
//  ---------------------------------------------------------------------------
// Example 2:
// The following polynomial is already x-distinguished
f = x6+y4+xy;
g,phi = makedistinguished(f,x);
g;                          // f is left unchanged
↦ x6+y4+xy
phi;                        // the transformation is the identity.
↦ phi[1]=x
↦ phi[2]=y
system("--random",randval);      // reset random generator
// ---------------------------------------------------------------------------
// Example 3:    // polynomials which cannot be transformed
// If p=char(basering)>0, then (p-1)*x^p*y + x*y^p factorizes completely
// in linear factors, since (p-1)*x^p+x equiv 0 on F_p. Hence,
// such polynomials cannot be transformed to a distinguished polynomial.
ring r3 = 3,(x,y),dp;
makedistinguished(2x3y+xy3,y);
↦    ? it could not be transform to a y-distinguished polynomial.
↦    ? leaving decomp.lib::makedistinguished
```

2 Index