Sparse polynomial arithmetic part II: A proposal to dramatically speed up polynomials in Maple.

Michael Monagan

CECM, Simon Fraser University

MOCAA M^3 workshop, May 7, 2008

Joint work with Roman Pearce, Simon Fraser University

Benchmarks

Dense Fateman problem (4 variables):

$$f = (1 + x + y + z + t)^{30} \quad g = f + 1$$

(stan.cecm.sfu.ca) Intel Core 2, 3.0 GHz, 64-bit

$46,376 \times 46,376 = 635,376$ terms $D = 0.000295 \quad 1/D = 3,385$	multiply $p = f \cdot g$	divide $q = p/f$
sdmp (packed)	54.720	68.160
sdmp (unpacked)	131.730	126.320
Trip v0.99 (rationals)	108.224	-
Pari 2.3.3 (w/ GMP)	512.184	283.445
Magma V2.14-7	679.070	610.620
Singular 3-0-4	1482.360	364.490
Maple 11	15986.169	13039.248

- f and g have 61 bit coefficients
- $h = f \cdot g$ has 128 bit coefficients

Benchmarks

Dense Fateman problem (3 variables):

$$f = (1 + x + y + z)^{30} \quad g = f + 1$$

(mado.cecm.sfu.ca) Intel Core 2 Duo, 2.4 GHz, 64-bit

$5,456 \times 5,456 = 39,711$ terms $D = 0.00133 \quad 1/D = 750$	multiply $p = f \cdot g$	divide $q = p/f$
sdmp (packed)	0.780	0.970
sdmp (unpacked)	1.330	1.510
Trip v0.99 (rationals)	1.870	-
Pari 2.3.3 (w/ GMP)	3.860	2.793
Magma V2.14-7	—	—
Singular 3-0-4	10.330	6.130
Maple 11	67.923	51.276

- f and g have 53 bit coefficients
- $h = f \cdot g$ has 112 bit coefficients

Benchmarks

Sparse 10 variables:

$$f = (x_1x_2 + x_2x_3 + x_3x_4 + x_4x_5 + x_5x_6 + x_6x_7 + x_7x_8 + x_8x_9 + x_9x_{10} + x_1x_{10} + x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + 1)^5$$

$$g = (x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2 + x_7^2 + x_8^2 + x_9^2 + x_{10}^2 + x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + 1)^5$$

(stan.cecm.sfu.ca) Intel Core 2, 3.0 GHz, 64-bit

$26,599 \times 36,365 = 19,631,157$ terms $D = 0.0203$ $\quad 1/D = 49.27$	multiply $p = f \cdot g$	divide $q = p/f$
sdmp (packed)	40.330	41.330
sdmp (unpacked)	175.970	162.370
Trip v0.99 (rationals)	221.910	-
Pari 2.3.3 (w/ GMP)	109.270	109.692
Magma V2.14-7	313.020	5744.600
Singular 3-0-4	655.250	206.600
Maple 11	14053.371	10760.364

First integration attempt

In expand.c in the kernel

```
if( (la-5)*(lb-5)>400 && la*lb>2500 )
    return( evalsysf("expand/bigprod","",New3(EXPSEQ,a,b)) );
```

In divide.c in the kernel

```
if( ID(b)==SUM && (LENGTH(a)-3)*(LENGTH(b)-3) > l*12000 )
    ta = evalsysf("expand/","bigdiv",New4(EXPSEQ,a,b,quo));
```

In the Maple library define

```
`expand/bigprod` := proc(a,b)
   sdmp:-AutoPack:-Multiply(a,b);
end:
`expand/bigdiv` := proc(a,b,q)
   sdmp:-AutoPack:-Divide(a,b,q);
end:
```

First integration attempt: Autopack

```
Divide := proc(f::polynom, g::polynom, q::name:=FAIL)
  local F, G, Q, d, s, vars, tord;
  vars := indets([f,g],'name');
  if not andmap(type, [f,g], 'polynom'('integer', vars)) then
     error "input must be polynomials with integer coefficients"
  end if;
  tord := 'grlex'(op(vars));
  d := degree(f,vars);
  if degree(g,vars) > d then return false; end if;
  d := max_pack(d);
  F := sdmp:-Import(f,tord,':-pack'=d);
  G := sdmp:-Import(g,tord,':-pack'=d);
  if q=FAIL then
    sdmp:-Divide(F,G);
  elif sdmp:-Divide(F,G,'s','Q');
    q := sdmp:-Export(Q);
    return true;
  else
    return false;
  end if
end proc;
```

First integration attempt: a factorization

```
    |\^/|     Maple 11 (X86 64 LINUX)
._|\|   |/|_. Copyright (c) Maplesoft
 \  MAPLE  /  All rights reserved. Maple is a trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
> X := s,t,x,y,z:
> f := randpoly([X],degree=10,terms=1000):
> g := randpoly([X],degree=10,terms=1000):
> h := expand(f*g):
> d := iquo(nops(f)*nops(g),nops(h)):
> printf("#f=%d  #g=%d  #h=%d  1/D=%5.2f\n",
>        nops(f),nops(g),nops(h),d);

#f=998  #g=991  #h=52023  1/D=19.00

> ft := time():

> factor(h): # Keith's Hensel lifting code (1984)

> time()-ft;
                                   120.260
> quit
```

First integration attempt: a factorization

polynomial multiplications: 996
trial divisions which failed: 61
divisions which succeeded: 117

```
> read "sdmp.mpl";
> `expand/bigprod` := proc(a,b) sdmp:-AutoPack:-Multiply(a,b); end:
> `expand/bigdiv` := proc(a,b,q) sdmp:-AutoPack:-Divide(a,b,q) end:
...
> st := time(): factor(h): time()-st;
```

26.998

First integration attempt: gcdex

```
>X := u,v,x,y,z;
>f := collect(x^5+randpoly([X],degree=4,terms=10),x);
>g := collect(x^4+randpoly([X],degree=3,terms=10),x);
>et := time(): gcdex(f,g,x); et := time()-et;
>ft := time(): gcdex(f,g,x,'s','t'); ft := time()-ft;
```

Old timings: et := 9.120 ft := 18.342
New timings: et := 0.263 ft := 1.190

First integration attempt: Overhead

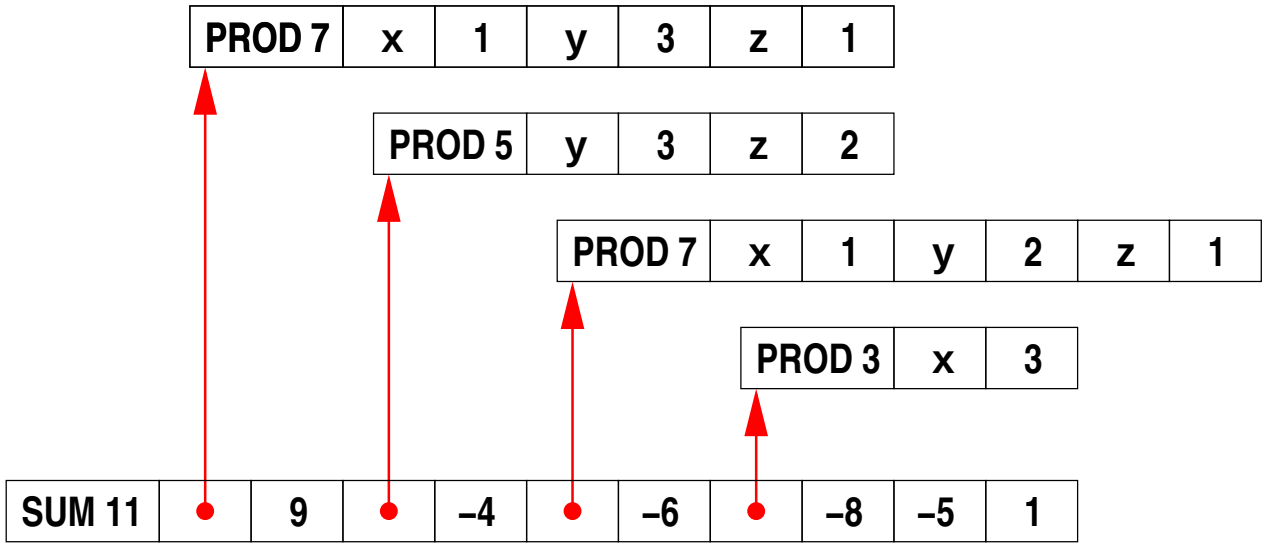
```
read "sdmp.mpl":
infolevel[sdmp] := 4;
interface(quiet=true);
X := [seq(x[i],i=1..10)];
for i from 2 to 10 do
  V := X[1..i];
  f := randpoly( V, degree=20-i, terms=1000 ):
  g := randpoly( V, degree=20-i, terms=1000 ):
  h := sdmp:-AutoPack:-Multiply(f,g);
  disp := evalf(nops(f)^2/nops(h));
  printf("#vars=%d  #f=%d  #h=%d  D=%4.1f\n",i,nops(f),nops(h),disp);
od:
```

First integration attempt: Overhead

```
#vars=4  #f=991  #h=10626  D=92.4
time: 0.020s   alg: 0.020s   io: 0.000s   real: 0.026s
time: 0.010s   export: 0.000s   simp: 0.010s   real: 0.012s

#vars=5  #f=988  #h=50876  D=19.2
time: 0.060s      alg: 0.060s   io: 0.000s   real: 0.067s
time: 0.070s   export: 0.010s   simp: 0.060s   real: 0.076s

#vars=6  #f=993  #h=169470  D= 5.8
time: 0.090s      alg: 0.080s   io: 0.010s   real: 0.094s
time: 0.330s   export: 0.030s   simp: 0.300s   real: 0.335s

#vars=7  #f=994  #h=334231  D= 3.0
time: 0.120s      alg: 0.110s   io: 0.010s   real: 0.119s
time: 0.760s   export: 0.080s   simp: 0.680s   real: 0.761s

#vars=8  #f=999  #h=440124  D= 2.3
time: 0.140s      alg: 0.110s   io: 0.030s   real: 0.132s
time: 1.080s   export: 0.120s   simp: 0.960s   real: 1.087s
```

Maple's sum of products representation.

$$9xy^3z - 4y^3z^2 - 6xy^2z - 8x^3 - 5$$



- small integer coefficients are immediate
- monomials are hashed, terms sorted by address

Maple's sum of products representation: Why is it so slow?

1: Because operations on PRODs takes 100s of cycles.

Example: consider multiplying $xy^2 \times (3x^2yz + 2xy - 3)$.

$$= xy^2 \boxed{\text{PROD} \mid x \mid 2 \mid y \mid 1 \mid z \mid 1}$$

$$= \boxed{\text{PROD} \mid x \mid 2 \mid y \mid 1 \mid z \mid 1 \mid x \mid 1 \mid y \mid 2}$$

$$= \boxed{\text{PROD} \mid x \mid 2 \mid y \mid 3 \mid z \mid 1 \mid x \mid 1 \mid y \mid 2}$$

$$= \boxed{\text{PROD} \mid x \mid 3 \mid y \mid 3 \mid z \mid 1 \mid x \mid 1 \mid y \mid 2}$$

I count sixteen C function calls to multiply and simpl.

Maple's sum of products representation: Why is it so slow?

2: Because large polynomials fill memory with PRODS.

This causes cache problems in expression walking e.g. in indets, degree, garbage collection, simpl table access, etc.

Maple's sum of products representation: Why is it so slow?

3: Maple sorts the terms in a SUM by address (good) using shellsort which is $O(n^{1.25})$ (okay) but it jumps through memory (bad).

```
if( l>50 ) {
    /* sort terms using shellsort */
    /* increment sequence is 3^k+1 */
    for( h=80; h<l; h=3*h+2 ) ;
    for( h/=3; h>1; h/=3 )
        for( i=h+1; i<l; i+=2 )
            for( j=i-h; j>0 && I(s[j])>I(s[j+h]); j-=h ) {
                r = A(s[j]);
                s[j] = s[j+h];
                s[j+h] = A2(r);
                r = A(s[j+1]);
                s[j+1] = s[j+h+1];
                s[j+h+1] = A2(r);
            }
```

Our data structure

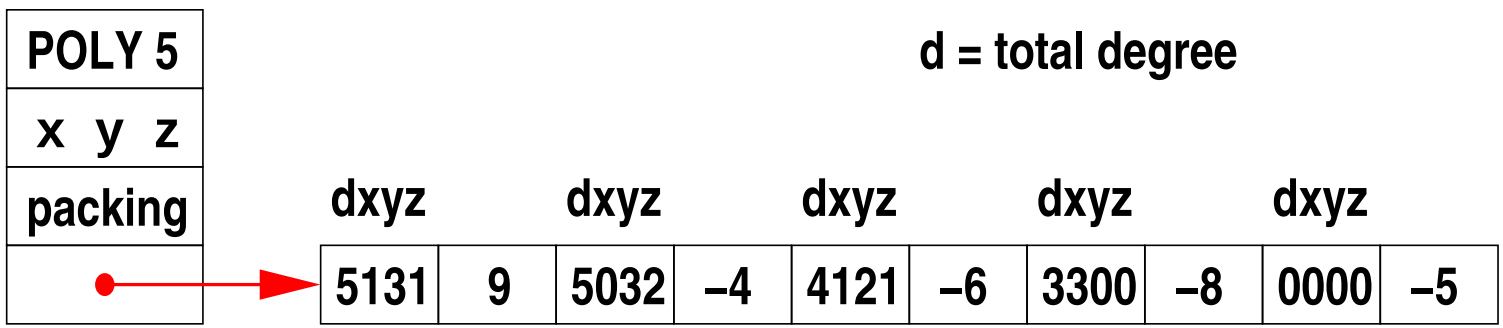
Packing for $x^i y^j z^k$ in graded lex order $x > y > z$:

In one machine word :

$i+j+k$	i	j	k

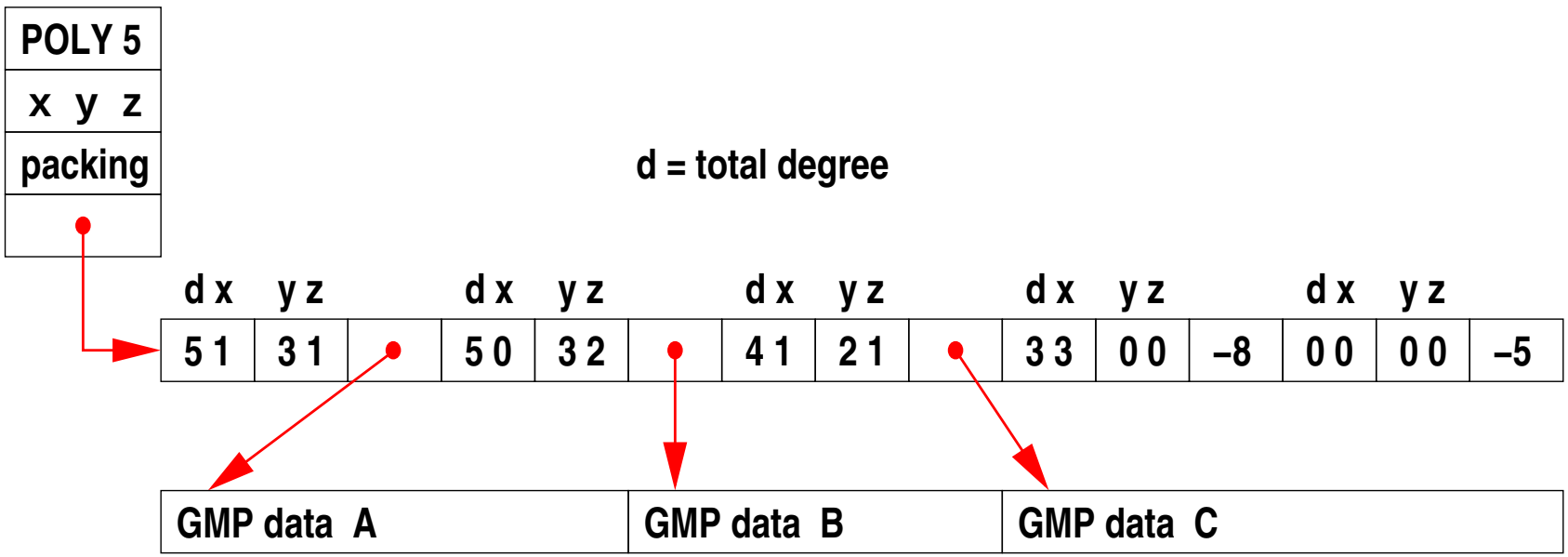
- monomial $>$ and $\times$ are one machine instruction.
- graded lex is good for polynomial division.

Packed array for: $9xy^3z - 4y^3z^2 - 6xy^2z - 8x^3 - 5$



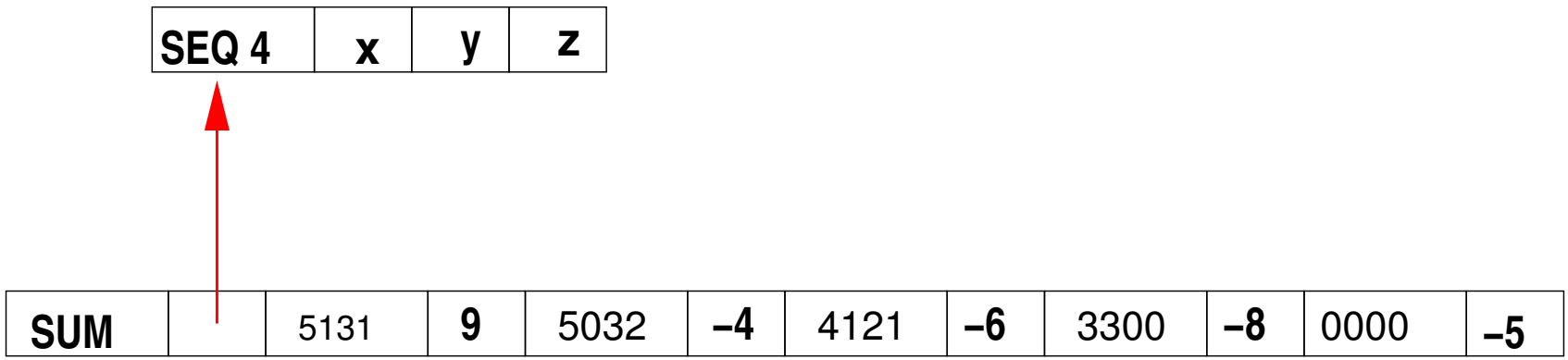
Our data structure: general case

$$Axy^3z - By^3z^2 - Cxy^2z - 8x^3 - 5$$



Our proposal: immediate monomials in grlex order

$$9xy^3z - 4y^3z^2 - 6xy^2z - 8x^3 - 5$$



Key: Packing is fixed by #variables.

So assuming grlex(x, y, z), to pack $x^i y^j z^k$ (3 variables), in 64 bits, we get need to store 4 integers, $(i + j + k, i, j, k)$.
Hence 16 bits per integer $\Longrightarrow$ $0 \leq i + j + k < 2^{16} = 65536$.

Our proposal: assumptions

- Polynomials which are created (e.g. by parsing) in the SUM of PRODs representation are simplified (in simpl) then "immediatized" in $O(Nm)$ time before hashing.
- We only pack expanded polynomials in names (functions?) whose monomials ALL pack into one machine word.
- To compute foo(f, g) if f is packed and g is not packed, then we convert f to SUM of PRODs and compute.
- If f and g are packed and indets$(f) \neq$ indets(g) and we can repack then make copy, repack if can, compute, simpl the result.

Our proposal: 64 bit verses 32 bit machines

	64 bit		32 bit	
#variables	#bits	max deg	#bits	max deg
1	32		16	
2	21		10	1023
3	16	65535	8	255
4	12	2047	6	63
5	10	1023	5	31
6	9	511	4	15
7	8	255	4	15
8	7	127	3	7
9	6	63	3	7
10	5	31	2	3
11	5	31	2	3
15	4	15	2	3
21	3	7	1	1
31	2	3	1	1
63	1	1	-	-

Our proposal: general comments

If f and g are polynomials in the same variables

- we can add and subtract f and g using a merge
- we can multiply f by g without overflow if $\deg(f) + \deg(g) < 2^b$.
- we can divide f by g without overflow if $\deg(f) < 2^b$ (needs grlex order)
- we can simpl the output polynomial result in $O(n)$ (a variable could drop out – this can be tested for in $O(n)$ time by bit-wise or of monomials)
- we can hash the result in $O(n)$
- If f has m variables and n terms, the storage is reduced from $2n + 1 + 2nm + 2m$ to $2n + 1$ for a gain of a factor of up to m.

Our proposal: the big gains

1. No overhead means we'll get the full factor of 100 gain in speed on large problems and a big gain on small and medium sized problems.
2. Because we eliminate PRODs, and since our multiplication and division algorithms run "inplace" we can multiply much larger polynomials.
3. Polynomials will appear to the user sorted, in descending order. Assuming we choose alphabetical order

input	output
$> 1 + 3x^2 - x;$	$3x^2 - x + 1$
$> \mathrm{expand}((x+y)^2);$	$x^3 + 3x^2y + 3xy^2 + 1$
$> x^3 + x + y^3 + 1;$	$x^3 + y^3 + x + 1$
$> 1 - x + xyz - 2x^2;$	$xyz - 2x^2 - x + 1$
$> -x - 2y + z + w = 0;$	$w - x - 2y + z = 0$
$> 1 - x_2x_1 + x_1^2;$	$x_1^2 - x_1x_2 + 1$

Our proposal: the small gains

There are other substantial gains in time and space that we get by using the new structure.

Let f be a polynomial, $N = \text{nops}(f)$, and $m = \text{nops}(\text{indets}(f))$.

Time from $O(Nm)$ to $O(1)$.

1. lcoeff(f);
2. indets(f); (possibly $O(m)$)
3. sign(f);
4. degree(f); (total degree of f)
5. expand(f); (f is already expanded)

Our proposal: more small gains

From $O(Nm)$ to $O(m)$.

```
6 > f; (evaluation of f at user level)
7 has(f,x);
8 type(f,polynom);
9 frontend(normal,f);
```

From $O(Nm)$ to $O(N+m)$.

```
10 degree(f,x);
11 coeff(f,x,1);
12 type(f,polynom(integer));
```

Key: $N = \text{nops}(f)$, $m = \text{nops}(\text{indets}(f))$.

Our proposal: more small gains

13 expand(x*f); from $O(Nm^2 + sort(N))$ to $O(N)$

15 $2f$; from $O(N)$ to $O(N)$.

16 diff(f,x); from $O(Nm^2 + sort(N))$ to $O(N + sort(N))$

17 eliminating monomials will shrink the SIMPL table

18 and reduce cache penalty for SIMPL table access

19 and result in faster garbage collection

Key: $N = \text{nops}(f)$, $m = \text{nops}(\text{indets}(f))$.

Our proposal: difficulties

```
> f := 1+x^2+y^2*x;
```

$$f := 1 + x^2 + y^2 x$$

```
> sort(f,[x,y],plex);
```

$$x^2 + x y^2 + 1$$

```
> sort(f,[x,y],tdeg);
```

$$x y^2 + x^2 + 1$$

What will we do for backwards compatibility?

Our proposal: solutions

- Propose default is grlex ordering.
  Defines the hash(f) for f with immediate monomials.
- For sort(f,[x,y,z],plex) we will sort in-place as follows. For $x^i y^j z^k$ we will do

$$\boxed{i+j+k}\,\boxed{i}\,\boxed{j}\,\boxed{k} \Longrightarrow \boxed{i}\,\boxed{j}\,\boxed{k}\,\boxed{i+j+k}.$$

  This guarantees that f is packed does not depend on the monomial ordering. If it did, we couldn't hash f consistently.
- We need to "tag" the ordering. Suggest store pointer to plex(x, y, z) instead of pointer to sequence (x, y, z) in the SUM dag.

Conclusion

The proposal is to introduce immediate monomials.

- Maple will be much faster than Magma and Singular.
- Maple will be **MUCH** faster than Mathematica.
- Maple will be able to manipulate much bigger polynomials.
- This will speed up Maple on the library test suite by better than 10%?
- Polynomials will be sorted.