# POLY : A new polynomial data structure for Maple.

Michael Monagan

Center for Experimental and Constructive Mathematics
Simon Fraser University
British Columbia
CANADA

ASCM 2012, Beijing,
October 26-28, 2012

This is joint work with Roman Pearce.

# Talk Outline

- Polynomial data structures in Maple and Singular are slow.
  Our new data structure POLY

# Talk Outline

- Polynomial data structures in Maple and Singular are slow.
  Our new data structure POLY
- Johnson's polynomial multiplication using a heap from 1973.
  Our parallelization of it.

# Talk Outline

- Polynomial data structures in Maple and Singular are slow.
  Our new data structure POLY
- Johnson's polynomial multiplication using a heap from 1973.
  Our parallelization of it.
- Maple 16 integration of POLY.
  A multiplication and factorization benchmark.

# Talk Outline

- Polynomial data structures in Maple and Singular are slow.
  Our new data structure POLY
- Johnson's polynomial multiplication using a heap from 1973.
  Our parallelization of it.
- Maple 16 integration of POLY.
  A multiplication and factorization benchmark.
- Maple 17 integration of POLY.
  New timings for same benchmark.

# Talk Outline

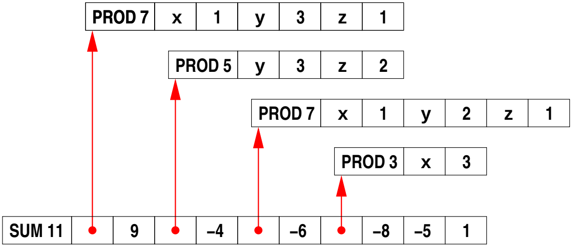
- Polynomial data structures in Maple and Singular are slow.
  Our new data structure POLY
- Johnson's polynomial multiplication using a heap from 1973.
  Our parallelization of it.
- Maple 16 integration of POLY.
  A multiplication and factorization benchmark.
- Maple 17 integration of POLY.
  New timings for same benchmark.
- Notes on integration into Maple 17 kernel.

# Talk Outline

- Polynomial data structures in Maple and Singular are slow.
  Our new data structure POLY
- Johnson's polynomial multiplication using a heap from 1973.
  Our parallelization of it.
- Maple 16 integration of POLY.
  A multiplication and factorization benchmark.
- Maple 17 integration of POLY.
  New timings for same benchmark.
- Notes on integration into Maple 17 kernel.
- Status of Maple 17 integration.

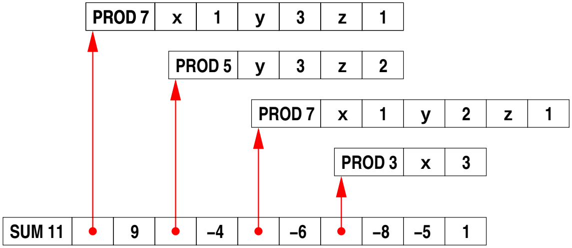
# Representations for $9\,xy^3z - 4\,y^3z^2 - 6\,xy^2z - 8\,x^3 - 5.$

Maple 16

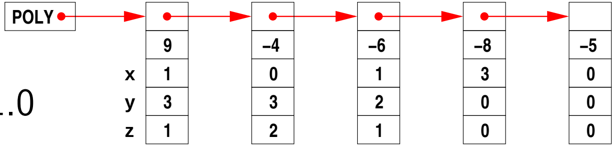


# Representations for $9\,xy^3z - 4\,y^3z^2 - 6\,xy^2z - 8\,x^3 - 5$.

Maple 16

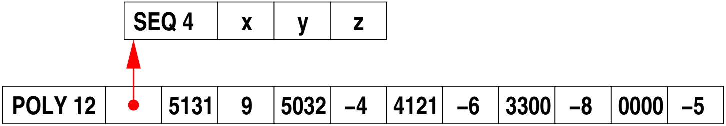



Singular 3.1.0




- Memory access is not sequential.
- Monomial multiplication costs O(100) cycles.

# Our representation $9\,xy^3z - 4\,y^3z^2 - 6\,xy^2z - 8\,x^3 - 5$.




Monomial encoding for graded lex order with $x > y > z$

Encodes $x^i y^j z^k$ in a single word $\boxed{d \mid i \mid j \mid k}$ where $d = i + j + k$.

Advantages

# Our representation $9\,xy^3z - 4\,y^3z^2 - 6\,xy^2z - 8\,x^3 - 5$.

| | | | |
|---|---|---|---|
| SEQ 4 | x | y | z |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| POLY 12 | • | 5131 | 9 | 5032 | –4 | 4121 | –6 | 3300 | –8 | 0000 | –5 |

Monomial encoding for graded lex order with $x>y>z$

Encodes $x^i y^j z^k$ in a single word $\boxed{d \mid i \mid j \mid k}$ where $d = i+j+k$.

Advantages

- It's more compact (2 words per term instead of 9).
- Memory access is sequential.
- Fewer objects to clutter tables.
- Monomial $>$ and $\times$ cost **one** instruction.

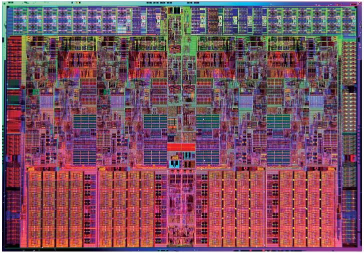
Let $f = f_1 + f_2 + \cdots + f_{\mathbf{n}}$ and $g = g_1 + g_2 + \cdots + g_{\mathbf{m}}$.
Compute $f \times g = f_{\mathbf{1}} \cdot g + f_{\mathbf{2}} \cdot g + \cdots + f_{\mathbf{n}} \cdot g$.

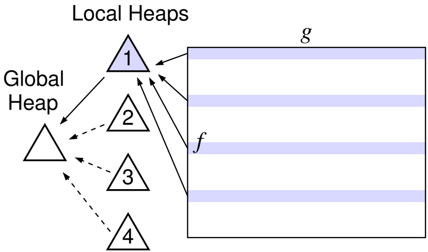
Naive merge: $O(mn^2)$ comparisons.
Johnson (1974) simultaneous $n$-ary merge (heap): $O(mn \log n)$.

Let $f = f_1 + f_2 + \cdots + f_{\mathbf{n}}$ and $g = g_1 + g_2 + \cdots + g_{\mathbf{m}}$.
Compute $f \times g = f_{\mathbf{1}} \cdot g + f_{\mathbf{2}} \cdot g + \cdots + f_{\mathbf{n}} \cdot g$.

Naive merge: $O(mn^2)$ comparisons.
Johnson (1974) simultaneous $n$-ary merge (heap): $O(mn \log n)$.

[MM, RP (2009)] parallel multiplication.
[MM, RP (2010)] parallel division.






Target architecture: Intel Core i7 (quad core)

# Old multiplication and factorization benchmark.

Intel Core i5 750 2.66 GHz (4 cores) — Times in seconds

| multiply | Maple 13 | Maple 16 1 core | Maple 16 4 cores | Magma 2.16-8 | Singular 3.1.0 | Mathematica 7 |
|---|---|---|---|---|---|---|
| $p_1 := f_1(f_1+1)$ | 1.60 | 0.053 | 0.029 | 0.30 | 0.58 | 4.79 |
| $p_4 := f_4(f_4+1)$ | 95.97 | 1.810 | 0.632 | 13.25 | 30.64 | 273.01 |
| factor | **Hensel lifting is mostly polynomial multiplication!** | | | | | |
| $p_1$ 12341 terms | 31.10 | 2.58 | 2.46 | 6.15 | 12.28 | 11.82 |
| $p_4$ 135751 terms | 2953.54 | 53.52 | 44.84 | 332.86 | 404.86 | 655.49 |

$f_1 = (1+x+y+z)^{20}+1$ 1771 terms

$f_4 = (1+x+y+z+t)^{20}+1$ 10626 terms

**Parallel speedup** for $f_4 \times (f_4+1)$ is $1.81 \;/\; .632 = \mathbf{2.86}\times$. **Why?**

# Maple 16 Integration of POLY

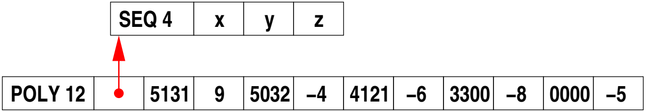
To expand sums $f \times g$ Maple calls `` `expand/bigprod(f,g)` ``
if $\#f > 2$ and $\#g > 2$ and $\#f \times \#g > 1500$.

```
`expand/bigprod` := proc(a,b)  # multiply two large sums
   if type(a,polynom(integer)) and type(b,polynom(integer)) then
     x := indets(a) union indets(b); k := nops(x);
     A := sdmp:-Import(a, plex(op(x)), pack=k);
     B := sdmp:-Import(b, plex(op(x)), pack=k);
     C := sdmp:-Multiply(A,B);
     return sdmp:-Export(C);
   else
   ...

`expand/bigdiv` := proc(a,b,q)  # divide two large sums
   ...
     x := indets(a) union indets(b); k := nops(x)+1;
     A := sdmp:-Import(a, grlex(op(x)), pack=k);
     B := sdmp:-Import(b, grlex(op(x)), pack=k);
   ...
```

# Make POLY the default representation in Maple.

If we can pack all monomials into one word use




| | |
|---|---|
| $O(1)$ | `degree(f); lcoeff(f); indets(f);` |
| $O(n)$ | `has(f,z); type(f,polynom(integer));` |
| $O(n+t)$ | `degree(f,x); expand(x*t); diff(f,x);` |

For $f$ with $t$ terms in $n$ variables.

# Almost everything is fast.

| command | Maple 16 | Maple 17 | speedup | notes |
|---|---|---|---|---|
| $\texttt{coeff}(f, x, 20)$ | 2.140 s | 0.005 s | 420x | terms easy to locate |
| $\texttt{coeffs}(f, x)$ | 0.979 s | 0.119 s | 8x | reorder exponents and radix |
| $\texttt{frontend}(g, [f])$ | 3.730 s | 0.000 s | $\rightarrow O(n)$ | looks at variables only |
| $\texttt{degree}(f, x)$ | 0.073 s | 0.003 s | 24x | stop early using monomial de |
| $\texttt{diff}(f, x)$ | 0.956 s | 0.031 s | 30x | terms remain sorted |
| $\texttt{eval}(f, x = 6)$ | 3.760 s | 0.175 s | 21x | use Horner form recursively |
| $\texttt{expand}(2 * x * f)$ | 1.190 s | 0.066 s | 18x | terms remain sorted |
| $\texttt{indets}(f)$ | 0.060 s | 0.000 s | $\rightarrow O(1)$ | first word in dag |
| $\texttt{op}(f)$ | 0.634 s | 2.420 s | 0.26x | has to construct old structur |
| `for t in f do` | 0.646 s | 2.460 s | 0.26x | has to construct old structur |
| $\texttt{subs}(x = y, f)$ | 1.160 s | 0.076 s | 15x | combine exponents, sort, me |
| $\texttt{taylor}(f, x, 50)$ | 0.668 s | 0.055 s | 12x | get coefficients in one pass |
| $\texttt{type}(f, polynom)$ | 0.029 s | 0.000 s | $\rightarrow O(n)$ | type check variables only |

For $f$ with $n = 3$ variables and $t = 10^6$ terms created by

```
f := expand(mul(randpoly(v,degree=100,dense),v=[x,y,z])):
```

# New multiplication and factorization benchmark

Intel Core i5 750 2.66 GHz (4 cores) Times in seconds

| multiply | Maple 16 1 core | Maple 16 4 cores | Maple 17 1 core | Maple 17 4 cores | Magma 2.16-8 | Singular 3.1.4 |
|---|---|---|---|---|---|---|
| $p_1 := f_1(f_1+1)$ | 0.053 | 0.029 | 0.042 | 0.017 | 0.30 | 0.57 |
| $p_4 := f_4(f_4+1)$ | 1.810 | 0.632 | 1.730 | 0.508 | 13.25 | 30.99 |
| factor | **Singular's factorization improved!** | | | | | |
| $p_1$ 12341 terms | 2.66 | 2.54 | 1.06 | 0.93 | 6.15 | 2.01 |
| $p_4$ 135751 terms | 56.68 | 44.06 | 26.43 | 16.17 | 332.86 | 61.85 |

$f_1 = (1+x+y+z)^{20}+1$ 1771 terms

$f_4 = (1+x+y+z+t)^{20}+1$ 10626 terms

**Parallel speedup** for $f_4 \times (f_4+1)$ is $1.730/0.508 = \mathbf{3.41}\times$.

# Profile for factor(p1);

Profile for `factor(p1);` Real time from 2.63s to 1.11s real.

| function | #calls | Maple 16 time | Maple 16 time% | New Maple time | New Maple time% |
|---|---|---|---|---|---|
| coeftayl | 216 | 0.999s | 36.96 | 0.270s | 22.39 |
| expand | 1934 | 0.561s | 20.75 | 0.375s | 31.09 |
| factor/diophant | 236 | 0.475s | 17.57 | 0.371s | 30.76 |
| divide | 419 | 0.267s | 9.88 | 0.055s | 4.56 |
| factor | 1 | 0.206s | 7.62 | 0.017s | 1.41 |
| factor/hensel | 1 | 0.140s | 5.18 | 0.075s | 6.22 |
| factor/unifactor | 2 | 0.055s | 2.03 | 0.043s | 3.57 |
| total: | 2809 | 2.703s | 100.00% | 1.206s | 100.00% |

The `coeftayl(f,x=a,k);` command is defined by `coeff(taylor(f,x=a,k+1),x,k);` and is computed via `eval(diff(f,x$k),x=a) / k!` which is 4x faster.

# Notes on the new integration for Maple 17.

Given a polynomial $f(x_1, x_2, ..., x_n)$, we store $f$ using POLY if

(1) $f$ is expanded and has integer coefficients,

(2) $d > 1$ and $t > 1$ where $d = \deg f$ and $t = \#$terms,

(3) we can pack all monomials of $f$ into one 64 bit word, i.e. if $d < 2^b$ where $b = \lfloor \frac{64}{n+1} \rfloor$

Otherwise we use the old sum-of-products representation.

# Notes on the new integration for Maple 17.

Given a polynomial $f(x_1, x_2, ..., x_n)$, we store $f$ using POLY if

(1) $f$ is expanded and has integer coefficients,

(2) $d > 1$ and $t > 1$ where $d = \deg f$ and $t = \#\text{terms}$,

(3) we can pack all monomials of $f$ into one 64 bit word, i.e. if $d < 2^b$ where $b = \lfloor \frac{64}{n+1} \rfloor$

Otherwise we use the old sum-of-products representation.

- The representation is invisible to the Maple user. Conversions are automatic.

# Notes on the new integration for Maple 17.

Given a polynomial $f(x_1, x_2, ..., x_n)$, we store $f$ using POLY if

(1) $f$ is expanded and has integer coefficients,

(2) $d > 1$ and $t > 1$ where $d = \deg f$ and $t = \#\text{terms}$,

(3) we can pack all monomials of $f$ into one 64 bit word, i.e. if $d < 2^b$ where $b = \lfloor \frac{64}{n+1} \rfloor$

Otherwise we use the old sum-of-products representation.

- The representation is invisible to the Maple user. Conversions are automatic.
- POLY polynomials will be displayed in sorted order.

# Notes on the new integration for Maple 17.

Given a polynomial $f(x_1, x_2, ..., x_n)$, we store $f$ using POLY if

(1) $f$ is expanded and has integer coefficients,

(2) $d > 1$ and $t > 1$ where $d = \deg f$ and $t = \#\text{terms}$,

(3) we can pack all monomials of $f$ into one 64 bit word, i.e. if $d < 2^b$ where $b = \lfloor \frac{64}{n+1} \rfloor$

Otherwise we use the old sum-of-products representation.

- The representation is invisible to the Maple user.
  Conversions are automatic.
- POLY polynomials will be displayed in sorted order.
- Packing is fixed by $n = \#\text{variables}$.

# Notes on the new integration for Maple 17.

Given a polynomial $f(x_1, x_2, \ldots, x_n)$, we store $f$ using POLY if

(1) $f$ is expanded and has integer coefficients,

(2) $d > 1$ and $t > 1$ where $d = \deg f$ and $t = \#$terms,

(3) we can pack all monomials of $f$ into one 64 bit word, i.e. if $d < 2^b$ where $b = \lfloor \frac{64}{n+1} \rfloor$

Otherwise we use the old sum-of-products representation.

- The representation is invisible to the Maple user. Conversions are automatic.
- POLY polynomials will be displayed in sorted order.
- Packing is fixed by $n = \#$variables.
- If $n = 8$, (3) $\implies$ we use $b = \lfloor 64/9 \rfloor = 7$ bits per exponent field hence POLY restricts $d < 128$.

# Current Work and Paper Content

We are trying to get it ready for Maple 17.

# Current Work and Paper Content

We are trying to get it ready for Maple 17.

- Avoid operations like indets(f,type) which unpack POLY.

# Current Work and Paper Content

We are trying to get it ready for Maple 17.

- Avoid operations like indets(f,type) which unpack POLY.
- Some external C libraries need POLY support

# Current Work and Paper Content

We are trying to get it ready for Maple 17.

- Avoid operations like indets(f,type) which unpack POLY.
- Some external C libraries need POLY support
- The different ordering has exposed hidden bugs.

The paper gives details on

# Current Work and Paper Content

We are trying to get it ready for Maple 17.

- Avoid operations like indets(f,type) which unpack POLY.
- Some external C libraries need POLY support
- The different ordering has exposed hidden bugs.

The paper gives details on

- Why we chose a graded ordering instead of lex order.

# Current Work and Paper Content

We are trying to get it ready for Maple 17.

- Avoid operations like indets(f,type) which unpack POLY.
- Some external C libraries need POLY support
- The different ordering has exposed hidden bugs.

The paper gives details on

- Why we chose a graded ordering instead of lex order.
- How we repack polynomials efficiently e.g. for `coeff`$(f, x, k)$.

# Current Work and Paper Content

We are trying to get it ready for Maple 17.

- Avoid operations like indets(f,type) which unpack POLY.
- Some external C libraries need POLY support
- The different ordering has exposed hidden bugs.

The paper gives details on

- Why we chose a graded ordering instead of lex order.
- How we repack polynomials efficiently e.g. for `coeff`$(f, x, k)$.
- A determinant benchmark showing a factor of **50** speedup.

# Current Work and Paper Content

We are trying to get it ready for Maple 17.

- Avoid operations like indets(f,type) which unpack POLY.
- Some external C libraries need POLY support
- The different ordering has exposed hidden bugs.

The paper gives details on

- Why we chose a graded ordering instead of lex order.
- How we repack polynomials efficiently e.g. for `coeff`$(f, x, k)$.
- A determinant benchmark showing a factor of **50** speedup.

Thank you for attending my talk.