Criteria for Compactness in the Design of Maple

Keith O. Geddes
Symbolic Computation Group
David R. Cheriton School of Computer Science
University of Waterloo, Waterloo, Ontario N2L 3G1 Canada

Presented at *CCA '09*
(Second Workshop on Compact Computer Algebra)
Grand Bend, Ontario
July 10, 2009

Abstract

The original paper on the design of Maple was published in 1983 under the following title: "*The design of Maple: A compact, portable and powerful computer algebra system*." As this title implies, compactness was a design goal for Maple from the beginning. In this presentation, we take a retrospective view of the design goals as presented in the early papers. The fundamental design concept was to develop a compact kernel for the computer algebra system, implemented as compiled code, which handles the most basic arithmetic and algebraic operations and defines an interpreter for the user-level Maple language. The mathematical power of the computer algebra system is achieved via a large library of algorithms written in the Maple user language, to be loaded and interpreted on an as-needed basis. The most significant improvements in (space and time) efficiency are achieved by the development and implementation of better algorithms for the various mathematical tasks, and the ease of writing algorithms in the user-level Maple language facilitates the timely incorporation of new algorithms.

In the Beginning

In the beginning (1980) there were two of us: Gaston Gonnet and Keith Geddes

Pre-Maple

• The computers were "large" (but lacking in speed and memory space!!)

The "red room" in the Math and Computer building at UW

Early Computer Algebra Systems

• The issue of ***memory size***
– in 1980, we were using a Honeywell time-sharing mainframe
– 50K words (i.e., 200K bytes) of memory usage was "very large"
– often wait overnight for processing to finish
– would lead to high charges to my research grant
– could not use more than a half-megabyte of memory

Example 1: How large do integers need to be?

• For example, compute the GCD of two polynomials

• In ALTRAN, max length for integers was 100 digits
(this is the maximum that could be requested!)

• 100 digits sounds "large enough" for many purposes

• *The Problem*: Intermediate steps of an algorithm may generate very large integers
(using the only GCD algorithms known at the time)

• The GCD algorithm used was a variation of the ancient Euclid algorithm (based on PRS, meaning Polynomial Remainder Sequences)

• The computation is ridiculously trivial in modern times!
(with ***modern algorithms*** as well as fast computers)

> $poly1 := 2500000\,x^4 - 487995500\,x^3 - 2442003501\,x^2 + 308523500\,x + 4504301$

$$poly1 := 2500000\,x^4 - 487995500\,x^3 - 2442003501\,x^2 + 308523500\,x + 4504301 \quad (1)$$

> $poly2 := 125000\,x^3 + 1110250\,x^2 + 2426470\,x + 2501$

$$poly2 := 125000\,x^3 + 1110250\,x^2 + 2426470\,x + 2501 \quad (2)$$

> $gcd(poly1, poly2)$

$$2501 + 500\,x \quad (3)$$

Note: The maximum length of integers above is 10 digits.

The old algorithm **`gcd/reduced`** (which uses a PRS algorithm) yields the same result. It even seems fast enough on modern computers.

However, notice the large integers that can be generated by intermediate calculations!

We very soon ran into GCD computations that could not be completed on a system that limits the size of integers to only 100 digits.

> $trace(`gcd/reduced`, `gcd/reduced/prs`)$

$$gcd/reduced, gcd/reduced/prs \quad (4)$$

> $`gcd/reduced`(poly1, poly2)$

```
{--> enter gcd/reduced, args = 2500000*x^4-487995500*x^3
-2442003501*x^2+308523500*x+4504301, 125000*x^3+1110250*
x^2+2426470*x+2501
{--> enter gcd/reduced/prs, args = 2500000*x^4
-487995500*x^3-2442003501*x^2+308523500*x+4504301,
125000*x^3+1110250*x^2+2426470*x+2501
```

$$vars := \{x\}$$

$$x := x$$

$$c := 2500000\,x^4 - 487995500\,x^3 - 2442003501\,x^2 + 308523500\,x + 4504301$$

$$d := 125000\,x^3 + 1110250\,x^2 + 2426470\,x + 2501$$

$$\alpha := 1$$

```
{--> enter gcd/reduced/prs, args = 2500000, 487995500
```

$$vars := \{\ \}$$

$$500$$

```
<-- exit gcd/reduced/prs (now in gcd/reduced/content) =
500}
{--> enter gcd/reduced/prs, args = 500, 2442003501
```

$$vars := \{\ \}$$

$$1$$

```
<-- exit gcd/reduced/prs (now in gcd/reduced/content) =
1}
```

$$c := 2500000\,x^4 - 487995500\,x^3 - 2442003501\,x^2 + 308523500\,x + 4504301$$

```
{--> enter gcd/reduced/prs, args = 125000, 1110250
```

$$vars := \{\ \}$$

$$250$$

```
<-- exit gcd/reduced/prs (now in gcd/reduced/content) =
250}
{--> enter gcd/reduced/prs, args = 250, 2426470
```

$$vars := \{\ \}$$

$$10$$

```
<-- exit gcd/reduced/prs (now in gcd/reduced/content) =
10}
{--> enter gcd/reduced/prs, args = 10, 2501
```

$$vars := \{\ \}$$

$$1$$

```
<-- exit gcd/reduced/prs (now in gcd/reduced/content) =
1}
```

$$d := 125000\,x^3 + 1110250\,x^2 + 2426470\,x + 2501$$

```
{--> enter gcd/reduced/prs, args = 1, 1
```

$$vars := \{\ \}$$

$$1$$

```
<-- exit gcd/reduced/prs (now in gcd/reduced/prs) = 1}
```

$$gam := 1$$

$$r := 229881134437500000 + 31891686562500000000\,x^2 + 159568174029375000000\,x$$

$$c := 125000\,x^3 + 1110250\,x^2 + 2426470\,x + 2501$$

$$true$$

$$\alpha := 15625000000$$

$$r := -10106422224337949216308593750000000000000000 - 20204762543658435058593750000000000000000000\,x$$

$$c := 229881134437500000 + 31891686562500000000\,x^2 + 159568174029375000000\,x$$

$$true$$

$$\alpha := 10170796718007430664062500000000000000000000$$

$$r := 0$$

```
{--> enter gcd/reduced/prs, args =
1293104802794139843750000000000,
6468110223576287498437500000000
```

$$vars := \{\ \}$$

$$25862096055882796875000000000$$

```
<-- exit gcd/reduced/prs (now in gcd/reduced/content) =
25862096055882796875000000000}
```

$$2501 + 500\,x$$

```
<-- exit gcd/reduced/prs (now in gcd/reduced) =
2501+500*x}
```

$$2501 + 500\,x$$

```
<-- exit gcd/reduced (now at top level) = 2501+500*x}
```

$$2501 + 500\,x \qquad (5)$$

```
>
```

- The point of the above trace of a computation: Large intermediate integers arise, much larger than the size of integers in the input and the output

Our 1983 paper on the design of Maple

The design criteria presented in the following bullet points come directly from our 1983 paper:

> **[Design83]** B.W. Char, K.O. Geddes, W.M. Gentleman and G.H. Gonnet, The design of Maple: A compact, portable and powerful computer algebra system. Appears in *Computer Algebra (Proceedings of EUROCAL'83)*, J.A. van Hulzen (ed.), Lecture Notes in Computer Science, No. 162, Springer-Verlag, Berlin, 1983, pp. 101—115.

Motivation for a New System

- user accessibility
  (including large classes of students)

Maple's specific goals:
- compactness
- powerful set of facilities for symbolic mathematics
- portability
- good user interface

Compact Size as a Design Goal

Compiled kernel is kept intentionally small

- *In 1983:* about 100K bytes (0.1 Mbytes)

- *Practical consequence:* Maple was very usable on personal computers with 1 Mbyte of memory

- Also, even today start-up time is quick

  - starting Maple 13 on Unix in non-GUI mode (the only mode in the 1980s!) starts instantaneously just as with the earliest Maple versions

  - about 0.8 Mbytes now

- the increased mathematical power resides in the external library

- starting Maple 13 on Windows (or whatever) with the full GUI interface is reasonably quick

Included in the compiled kernel:

• Interpreter for the user-level Maple language

• Basic arithmetic and algebraic operations
(numerical, polynomial, series)

• Basic simplification

• Tables and arrays

• Print routines
(in 1983: two-dimensional display)

• Fundamental functions
(***coeff, degree, subs, map, igcd, lcoeff, op, divide,*** etc.)

• Functions with a "core" in the kernel and an interface to Maple routines
(***diff, expand, taylor, type,*** etc.)

Not included in the kernel

• High-level mathematical functions are entirely in the external library (written in the Maple language)
(***gcd, factor, int, sum, solve,*** etc.)

Achieving compactness

• The use of appropriate data structures
- implemented as *dynamic vectors*

• The use of a viable file system
- *"you only pay for what you use"*
- assumes a tree-structured directory system and variable-length records

• Avoiding a large run-time support system

• A policy of treating main memory as a scarce resource

• The choice of the BCPL family of systems implementation languages

A rich set of data structures

- for programming-language statements
  e.g., *assignment, if, read,* etc.

- for standard objects
  e.g., *integer, rational number, list, set,* etc.

- for mathematical objects
  e.g., *sum, logical and, series, equation,* etc.

- for other expressions
  e.g., *procedure definitions, tables,* etc.

Example 2: A Maple data structure

Reference: **[Textbook92]**

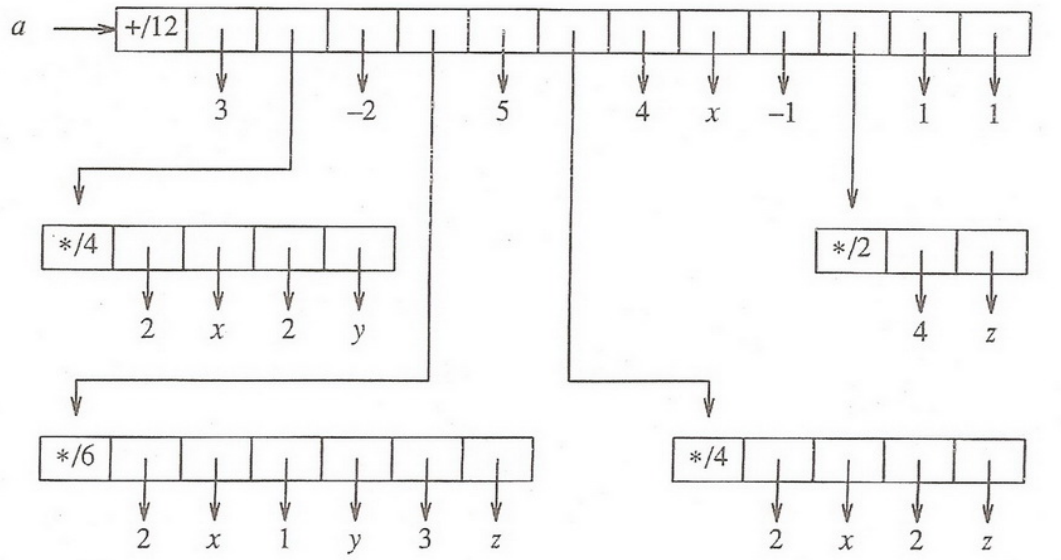


Figure 3.3. A dynamic array representation.

Example 3.4. Let $a(x,y,z) \in \mathbb{Z}[x,y,z]$ be the polynomial

$$a(x,y,z) = 3x^2y^2 - 2x^2yz^3 + 5x^2z^2 + 4x - z^4 + 1$$

Advantages of dynamic vectors

• only one level of abstraction above the system level of objects

• therefore, simpler and more efficient code
  - compared with having an intermediate "artificial" structure such as lists

• data structures related to the language
  - i.e., to the productions in the grammar

• direct access to each component of a structure

• structures are more compact
  - compared with linked lists

Computational Power through Libraries of Functions

• compactness not achieved at the expense of the power of the system

• a large number of functions provided in the external library

• functions are stored as a load module in a quick-loading, expression-tree representation of the procedure definition

• library functions are interpreted

• critical functions go into the compiled Maple kernel

(Space and Time) Efficiency via Better Algorithms

The following points appear in our 1984 paper:

> **[Design84]** B.W. Char, G.J. Fee, K.O. Geddes, G.H. Gonnet, M.B. Monagan and S. M. Watt, On the design and performance of the Maple system. *Proceedings of the 1984 MACSYMA Users' Conference*, V. Ellen Golden (ed.), General Electric, Schenectady, New York, 1984, pp. 199—219.

• A fundamental design criterion for the Maple system is that *space is more critical than time.*

• In addition to the issue of the size of the compiled kernel, the *run-time* consumption of data space and processor time is of great importance in computer algebra computations.

• The most significant gains in (space and time) efficiency are achieved via better algorithms.

• Contributions to better algorithms are facilitated by expressing the algorithms in the user-level Maple language
(rather than coding in a system implementation language such as C or Lisp).

- When designing algorithms, often there are variations of an algorithm which trade off space consumption versus time consumption. In Maple development, we used the following measure: $Cost = Space^2 \cdot Time$.
- Specifically, when choosing between two algorithms we would prefer the algorithm with lower *Cost* as defined by the above measure.

Algorithms for GCD computation

Reference: **[Textbook92]**

Note: The essential ideas in Algorithm 2.1, as it applies to positive integers, date back to Euclid, circa 300 B.C.

Algorithm 2.1. Euclidean Algorithm.

procedure Euclid(a, b)

Compute $g = \text{GCD}(a, b)$, where a and b
are from a Euclidean domain D .

$c \leftarrow \text{n}(a)$; $d \leftarrow \text{n}(b)$

while $d \neq 0$ **do** {
    $r \leftarrow \text{rem}(c, d)$
    $c \leftarrow d$
    $d \leftarrow r$ }

$g \leftarrow \text{n}(c)$
return(g)

end

- For polynomial GCD computation one can apply Algorithm 2.1, with some modifications to

handle the case of multivariate polynomials, yielding a PRS (*Polynomial Remainder Sequence*) algorithm.

- More precisely, there are several variations of PRS algorithms for polynomial GCD computation.
- In the early days of computer algebra systems, the only known GCD algorithms for polynomials were the PRS algorithms.
- Since the 1970s, several better algorithms have been developed for polynomial GCD computation. The new algorithms are generally based on *reduction* via homomorphisms followed by some type of *lifting*.
- Two new algorithms developed by our Maple group are in the references **[GCDHEU89]** and **[HeuAlg89]**.

Example 3: A univariate example

```
> poly3 := 7938000*x^9+56348885*x^6+103211248*x^3+19208000;
```

$$poly3 := 7938000\,x^9 + 56348885\,x^6 + 103211248\,x^3 + 19208000 \quad (6)$$

```
> poly4 := 47437425*x^9+888761683*x^6+4980881360*x^3+9739640000;
```

$$poly4 := 47437425\,x^9 + 888761683\,x^6 + 4980881360\,x^3 + 9739640000 \quad (7)$$

The computation is trivial using modern algorithms.

```
> gcd(poly3, poly4);
```

$$1 \quad (8)$$

Applying the "Reduced PRS Algorithm", note the extremely large integers which arise prior to the final answer.

```
> trace(`gcd/reduced`, `gcd/reduced/prs`);
```

$$gcd/reduced,\ gcd/reduced/prs \quad (9)$$

```
> `gcd/reduced`(poly3, poly4);
{--> enter gcd/reduced, args = 7938000*x^9+56348885*
x^6+103211248*x^3+19208000, 47437425*x^9+888761683*
x^6+4980881360*x^3+9739640000
{--> enter gcd/reduced/prs, args = 7938000*x^9+56348885*
x^6+103211248*x^3+19208000, 47437425*x^9+888761683*
x^6+4980881360*x^3+9739640000
```

$$vars := \{x\}$$

$$x := x$$

$$c := 7938000\,x^9 + 56348885\,x^6 + 103211248\,x^3 + 19208000$$

$$d := 47437425\,x^9 + 888761683\,x^6 + 4980881360\,x^3 + 9739640000$$

$$\alpha := 1$$

```
{--> enter gcd/reduced/prs, args = 7938000, 56348885
```

$$vars := \{\ \}$$

$$5$$

```
<-- exit gcd/reduced/prs (now in gcd/reduced/content) =
5}
{--> enter gcd/reduced/prs, args = 5, 103211248
```

$$vars := \{\ \}$$

$$1$$

```
<-- exit gcd/reduced/prs (now in gcd/reduced/content) =
1}
```

$$c := 7938000\,x^9 + 56348885\,x^6 + 103211248\,x^3 + 19208000$$

```
{--> enter gcd/reduced/prs, args = 47437425, 888761683
```

$$vars := \{\ \}$$

$$1$$

```
<-- exit gcd/reduced/prs (now in gcd/reduced/content) =
1}
```

$$d := 47437425\,x^9 + 888761683\,x^6 + 4980881360\,x^3 + 9739640000$$

```
value remembered (in gcd/reduced/prs): gcd/reduced/prs
(1, 1) -> 1
```

$$gam := 1$$

$$r := -4381944233632875\,x^6 - 34642160399523600\,x^3 - 76402084260600000$$

$$c := 47437425\,x^9 + 888761683\,x^6 + 4980881360\,x^3 + 9739640000$$

$$true$$

$$\alpha := 47437425$$

$$r := 288425614657570901224196347286707807434999826466846020324434375000000000 + 340469272451389547542643671456402229436287652208788563393577000000000000\,x^3$$

$$c := -4381944233632875\,x^6 - 34642160399523600\,x^3 - 76402084260600000$$

$$true$$

$$\alpha := 368695116300057130801197240417866121885729634081329336181640625$$

$$r := -258465298203077730062757650360326508392291938510463338506908960677288328787640505809122339860904576561642656515706908619960536820652915497603775895222377479326188614806947389196075448403825086503907466566207822168448413844703287500000000000000000000000000000000000000000000$$

$$1$$

```
<-- exit gcd/reduced/prs (now in gcd/reduced) = 1}
```

$$1$$

```
<-- exit gcd/reduced (now at top level) = 1}
```

$$1 \tag{10}$$

>

Example 4: A multivariate example

> $poly5 := x^4y^4 + 2x^3y^3z + 4x^3y^3 + 2x^2y^2z^2 + 4x^2y^2z + 5x^2y^2 + 2xyz^3 + 4xyz^2 + 2xyz + 4xy + z^4 + 4z^3 + 5z^2 + 4z + 4$

$$poly5 := x^4y^4 + 2x^3y^3z + 4x^3y^3 + 2x^2y^2z^2 + 4x^2y^2z + 5x^2y^2 + 2xyz^3 + 4xyz^2 + 2xyz + 4xy + z^4 + 4z^3 + 5z^2 + 4z + 4 \tag{11}$$

> $poly6 := x^4y^2 - x^2y^4 + x^2y^2z^2 - x^2y^2 + x^2z^2 + x^2 - y^2z^2 - y^2 + z^4 - 1$

$$poly6 := x^4y^2 - x^2y^4 + x^2y^2z^2 - x^2y^2 + x^2z^2 + x^2 - y^2z^2 - y^2 + z^4 - 1 \tag{12}$$

The computation is trivial using modern algorithms.

> $gcd(poly5, poly6)$

$$z^2 + 1 + x^2y^2 \tag{13}$$

>

Applying the "Reduced PRS Algorithm", turn off tracing to suppress the multi-page output!

> $untrace(`gcd/reduced`, `gcd/reduced/prs`)$

$$gcd/reduced, gcd/reduced/prs \tag{14}$$

> $`gcd/reduced`(poly5, poly6)$

$$z^2 + 1 + x^2y^2 \tag{15}$$

>

By tracing the computation, we find that one of the intermediate remainders generated is as follows.

$$r := -8y^{10} + 16y^8 - 8y^{10}z - 3y^{12}z^4 - 11y^{12}z^2 + 2y^{10}z^6 - 8y^{12}z^3 + 8y^{10}z^5 - 2y^{10}z^2 - 8y^{12}z - 7y^{14}x^2 - 8y^{12}x^2 + 8y^{10}z^4 + 16y^{10}x^2 + 25y^8z^4 + 40y^8z^3 + 32y^8z + y^{18}x^2 + 2y^{16}x^2 + y^{16}z^2 - 2y^{14}z^4 + 8y^8z^5 + y^8z^6 + y^{12}z^6 + 24y^{10}x^2z^2 + 32y^{10}x^2z - 2y^{16}x^2z^2 + 8y^{10}z^3x^2 + y^{10}z^4x^2 + y^{14}z^4x^2 + 40y^8z^2 - 8y^{14}zx^2 - 8y^{12}zx^2 + 6y^{12}z^2x^2 - 4y^{14}z^2x^2 + 8y^{12}z^3x^2 + 2y^{12}z^4x^2 - 7y^{12} + y^{16} + 2y^{14}$$

One again, expression swell occurs with the result that too much memory space is consumed compared with better algorithms.

>

Conclusion

- Computer Algebra algorithms may consume, at runtime, excessive amounts of memory space (usually implying excessive processing time as well!).

• Using better algorithms is a crucial aspect of achieving space and time efficiency.

1981-1985: A few pictures

Michael Monagan and Greg Fee

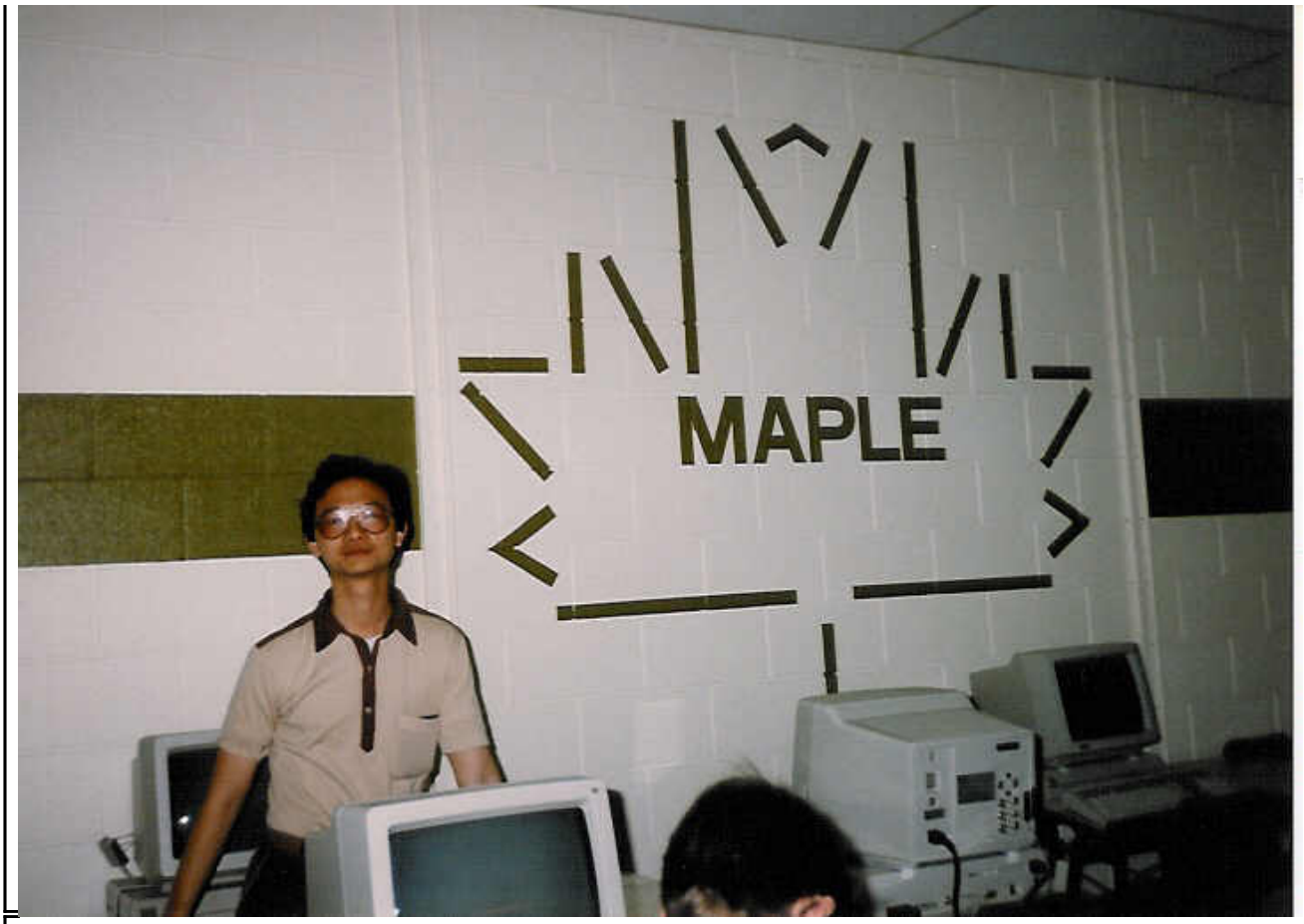
• Michael Monagan arrived at UW (from New Zealand) as a graduate student in 1982.

• By the end of the 1980s when we needed to calculate the contributions to Maple code by various authors, Michael was one of the "top three" contributors (Gaston, Keith and Michael).

• Michael's contributions have been at the level of the Maple kernel code as well as external library code.

• Greg Fee and Benton Leong were hired to work on the Maple project about 1981 and remained with the project for many years.

Benton Leong and Howard Johnson



A good view of the "ASCII Maple logo"

- Credit for inventing the "ASCII Maple logo" goes to Stephen Watt, who completed his PhD degree under my supervision in the early 1980s.

Stan Devitt, Stephen Watt, Bruce Char, Benton Leong, Keith Geddes
(in Linz, Austria for the 1985 Eurocal conference)

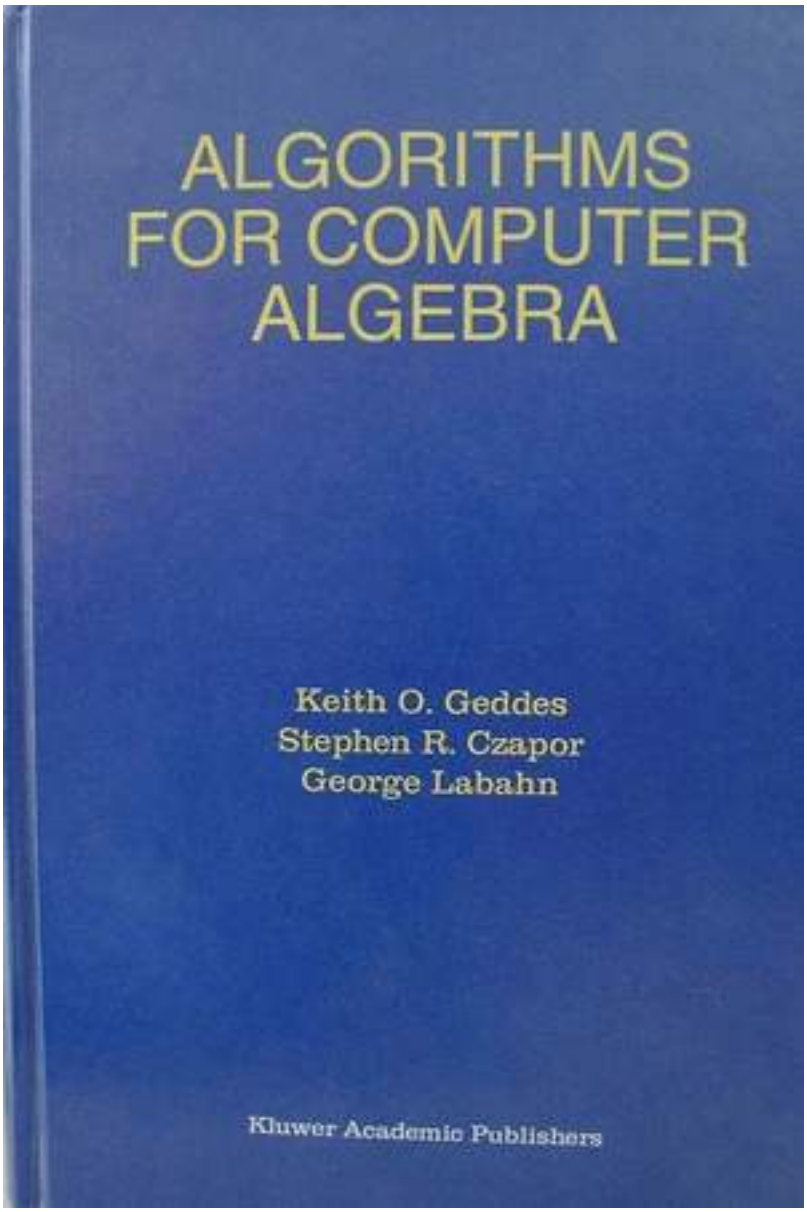
A geeky picture of me in Linz, Austria (1985)

1986-2006: More pictures

ISSAC '88 in Rome -- the first to be named "ISSAC"

Rome 1988

Bruce Char and Mark Mutrie, Maple Retreat 1989



The good book is finally completed in 1992!

Maple booth, NCTM 1992, Quebec City

A view of the competition, NCTM 1992, Quebec City

Stephen Watt and I with hosts, Kiev 1993

Kiev 1993

The vodka banquet, Kiev 1993

W. Kahan, K. Geddes, D. Jeffrey, G. Labahn, Ha Le, Maple Retreat 1994

Rob Corless family, Maple Retreat 1994

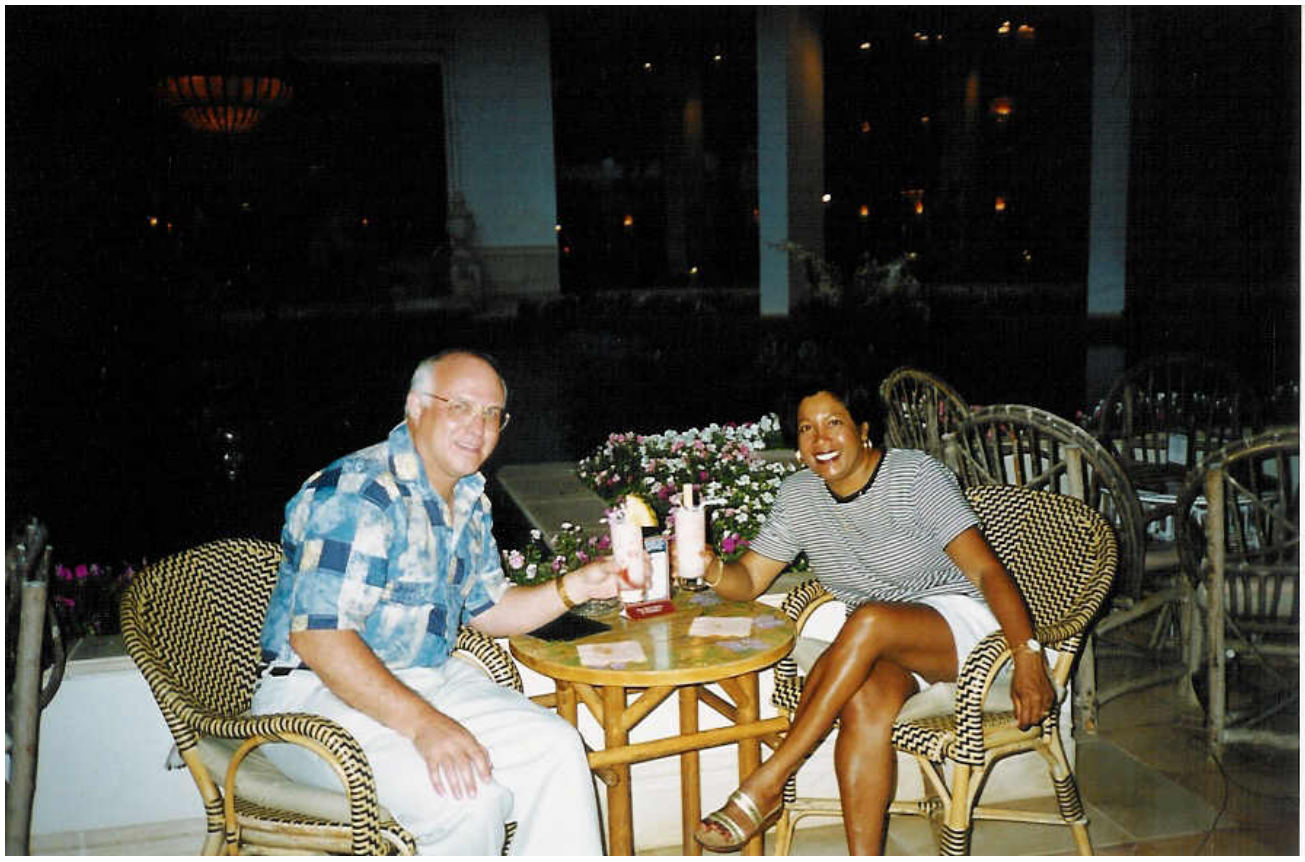
G. Gonnet, B. Salvy, M. Bronstein, Maple Retreat 1994

Erich Kaltofen with Allan Bonadio and ??, Maple Retreat 1994

Group picture, Maple Retreat 1994

Hard at work at the Maple Retreat, Sparrow Lake, Ontario, 1997

A view outside the ISSAC '97 conference, Maui, Hawaii



References

[Design83] B.W. Char, K.O. Geddes, W.M. Gentleman and G.H. Gonnet, The design of Maple: A compact, portable and powerful computer algebra system. Appears in *Computer Algebra (Proceedings of EUROCAL'83)*, J.A. van Hulzen (ed.), Lecture Notes in Computer Science, No. 162, Springer-Verlag, Berlin, 1983, pp. 101—115.

[Design84] B.W. Char, G.J. Fee, K.O. Geddes, G.H. Gonnet, M.B. Monagan and S.M. Watt, On the design and performance of the Maple system. *Proceedings of the 1984 MACSYMA Users' Conference*, V. Ellen Golden (ed.), General Electric, Schenectady, New York, 1984, pp. 199—219.

[GCDHEU89] B.W. Char, K.O. Geddes and G.H. Gonnet, GCDHEU: Heuristic polynomial GCD algorithm based on integer GCD computation. *J. Symbolic Computation*, 7, 1989, pp. 31—48.

[HeuAlg89] K.O. Geddes, G.H. Gonnet and T.J. Smedley, Heuristic methods for operations with algebraic numbers. Appears in *Symbolic and Algebraic Computation*, P. Gianni (ed.), Lecture Notes in Computer Science, No. 358, Springer-Verlag, Berlin, 1989, pp. 475—480.

[Textbook92] K.O. Geddes, S.R. Czapor and G. Labahn, *Algorithms for Computer Algebra*. Kluwer Academic Publishers, Boston, 1992, 585 pages.