Generic, open and powerful

Marc Moreno Maza

September 1, 2006

An introductory example

```
Z := Integer
   (1)  Integer
                                                    Type: Domain
                                                    Time: 0 sec
Q := Fraction Z
   (2)  Fraction Integer
                                                    Type: Domain
                                                    Time: 0 sec
P := Polynomial Q
   (3)  Polynomial Fraction Integer
                                                    Type: Domain
                                                    Time: 0 sec
K := Fraction P
   (4)  Fraction Polynomial Fraction Integer
                                                    Type: Domain
                                                    Time: 0 sec
```

```
M := SquareMatrix(4, K)
   (5)  SquareMatrix(4,Fraction Polynomial Fraction Integer)
                                                    Type: Domain
                                                    Time: 0 sec

quaternion(a: K, b: K, c: K, d: K): M ==
   matrix ([[d ,  a ,  b ,  c],
            [-a ,  d ,  -c ,  b],
            [-b ,  c ,  d ,  -a],
            [-c ,  -b ,  a ,  d ]])

   Function declaration quaternion : (Fraction Polynomial
      Fraction Integer,Fraction Polynomial Fraction Integer,
      Fraction Polynomial Fraction Integer,Fraction Polynomial
      Fraction Integer) -> SquareMatrix(4,Fraction Polynomial
      Fraction Integer) has been added to workspace.
```

```
q := quaternion(1,2,3,x/y)
         + x                  +
         | -    1    2    3 |
         | y                  |
         |                    |
         |      x             |
         |- 1   -   - 3   2 |
         |      y             |
   (7)   |                    |
         |           x        |
         |- 2   3    -   - 1|
         |           y        |
         |                    |
         |                x |
         |- 3  - 2   1    - |
         +                y +
```

```
recip(q)

       +   1             1  2         1  2          3  2  +
       |  -- x y         -- y          - y          -- y   |
       |  14             14            7            14     |
       |----------  - ----------  - ----------  - ----------|
       | 2    1  2     2    1  2     2    1  2     2    1  2|
       |y  + -- x     y  + -- x     y  + -- x     y  + -- x |
       |     14            14            14            14   |
       |                                                    |
       |    1  2         1              3  2          1  2   |
       |   -- y          -- x y         -- y          - y    |
       |   14            14             14            7      |
       |----------   ----------    ----------   - ----------|
       | 2    1  2    2    1  2     2    1  2      2    1  2|
       |y  + -- x    y  + -- x     y  + -- x      y  + -- x |
       |     14           14             14             14   |
  (8)  |                                                    |
       |   1  2           3  2          1              1  2   |
       |   - y           -- y          -- x y         -- y    |
       |   7             14            14             14      |
       |----------  - ----------   ----------    ---------- |
       | 2    1  2     2    1  2    2    1  2     2    1  2 |
       |y  + -- x     y  + -- x    y  + -- x     y  + -- x  |
       |     14            14            14             14   |
       |                                                    |
       |    3  2         1  2            1  2         1     |
       |   -- y          - y            -- y         -- x y  |
       |   14            7              14           14      |
       |----------   ----------   - ----------   ---------- |
       | 2    1  2    2    1  2      2    1  2    2    1  2 |
       |y  + -- x    y  + -- x      y  + -- x    y  + -- x  |
       +     14            14             14           14   +
  Type: Union(SquareMatrix(4,Fraction Polynomial Fraction Integer),...)
                Time: 0.05 (EV) + 0.02 (OT) + 0.01 (GC) = 0.08 sec
```

```
quaternion(1,2,3,q)
    Cannot convert from type SquareMatrix(4,Fraction
    Polynomial Fraction Integer) to Fraction Polynomial
```

$$
\begin{bmatrix}
x & & & \\
- & 1 & 2 & 3 \\
y & & & \\
 & & & \\
 & x & & \\
-1 & - & -3 & 2 \\
 & y & & \\
 & & & \\
 & & x & \\
-2 & 3 & - & -1 \\
 & & y & \\
 & & & \\
 & & & x \\
-3 & -2 & 1 & - \\
 & & & y
\end{bmatrix}
$$

A brief history

• The `Scratchpad` projects were developed since 1971 by IBM under the direction of **Richard Jenks.**

• `Scratchpad 2` was considered as a research platform for developing new ideas in computational mathematics.

• From 1991, ALDOR, an extension language for `Scratchpad 2` was developed under the lead of **Stephen Watt.**

• In 1990s, the `Scratchpad 2` project was renamed to AXIOM. It was sold to the **Numerical Algorithms Group (NAG)** and became a commercial system.

• AXIOM was withdrawn from the market in October, 2001.

• **NAG** agreed to release AXIOM and ALDOR as **free software**.

• Today, the efforts of the AXIOM community are coordinated by **Tim Daly.** Those around ALDOR are organized at `ORCCA`.

More recently

- Released under Modified BSD License in Sept. 2002.
- GNU Common Lisp port from Codemist Common Lisp
- Literate rewrite; Algebra code released in Sept. 2003
- Electronic Book released in April 2004
- Graphics released in Sept. 2004
- Browser released in Dec. 2004
- Feature complete release in Feb. 2005
- Introducing fast arithmetic and parallelism in AXIOM/ALDOR (Yuzhen Xie, Xin Li and Akpodigha Filatei)

A few data

- Approximately 3.6 million lines of source, including documentation.
- The whole system takes about 8 hours to build on a 1Ghz machine.
- Free and Open Source software maintained on several servers:
  - **savannah**, by courtesy of the Free Software Foundation,
  - **sourceforge**, by courtesy of the Open Source Technology Group,
  - **axiom-developer.org**, by Tim Daly, with support from the Center for Algorithms and Interactive Scientific Software (CAISS) at the City College of New York.
- AXIOM has 82 registered developers and 22 of them have write access to the source code.
- Direct contributions, over 35 years, of approximately 164 people.

AXIOM **contributors**

Cyril Alberga	Roy Adler	Richard Anderson
George Andrews		
Henry Baker	Stephen Balzac	Yurij Baransky
David R. Barton	Gerald Baumgartner	Gilbert Baumslag
Fred Blair	Mark Botch	Alexandre Bouyer
Peter A. Broadbery	Martin Brock	Manuel Bronstein
Florian Bundschuh	William Burge	
Robert Caviness	Bruce Char	Cheekai Chin
David V. Chudnovsky	Gregory V. Chudnovsky	Josh Cohen
Don Coppersmith	George Corliss	Robert Corless
Gary Cornell		
Timothy Daly Sr.	Timothy Daly Jr.	James H. Davenport
Jean Della Dora	Michael Dewar	Claire DiCrescendo

Sam Dooley
Brian Dupee

Robert Edwards

Richard Fateman
Brian Ford
Timothy Freeman

Marc Gaetano
Holger Gollan
Stephen Gortler
James Griesmer

Steve Hague
Martin Hassner

Pietro Iglio

Lionel Ducos
Dominique Duval

Lars Erickson

Bertfried Fauser
Albrecht Fortenbacher
Korrinn Fu

Rudiger Gebauer
Teresa Gomez-Diaz
Johannes Grabmeier
Vladimir Grinberg

Vilya Harvey
Henderson

Martin Dunstan

Stuart Feldman
George Frances

Patricia Gianni
Laureano Gonzalez-
Matt Grayson
Oswald Gschnitzer

Satoshi Hamaguchi

Richard Jenks

Grant Keady	Tony Kennedy	Klaus Kusche
Bernhard Kutzler		
Larry Lambe	Frederic Lehobey	Michel Levaud
Xin Li	Rudiger Loos	Michael Lucks
Richard Luczak		
Camm Maguire	Bob McElrath	Michael McGettrick
Ian Meikle	David Mentre	Victor S. Miller
Gerard Milmeister	Mohammed Mobarak	H. Michael Moeller
Michael Monagan	Marc Moreno Maza	Scott Morrison
Mark Murray		
William Naylor	C. Andrew Neff	John Nelder
Godfrey Nolan	Arthur Norman	

Michael O’Connor

Julian A. Padget	Bill Page	Susan Pelzel
Michel Petitot	Didier Pinchon	

Claude Quitte

Norman Ramsey	Michael Richardson	Renaud Rioboo
Jean Rivlin	Simon Robinson	Michael Rothstein

Philip Santas	Alfred Scheerhorn	William Schelter
Gerhard Schneider	Martin Schoenert	Marshall Schor
Fritz Schwarz	Nick Simicich	William Sit
Jonathan Steinbach	Christine Sundaresan	Robert Sutor
Moss E. Sweedler	Eugene Surowitz	

James Thatcher	Dylan Thurston	Barry Trager

Themos T. Tsikas

Bernhard Wall	Stephen Watt	Juergen Weiss
M. Weller	Mark Wegman	James Wen
Thorsten Werther	John M. Wiley	Berhard Will
Clifton J. Williamson	Shmuel Winograd	Robert Wisbauer
Waldemar Wiwianka	Knut Wolf	

David Yun

Richard Zippel	Bruno Zuercher	Dan Zwillinger

AXIOM: **Genericity**

Being able to implement mechanisms for producing new algebraic structures from existing ones:

(1) $R = Z$,

(2) $n \longmapsto Z/nZ$,

(3) $(R, X) \longmapsto R[X]$,

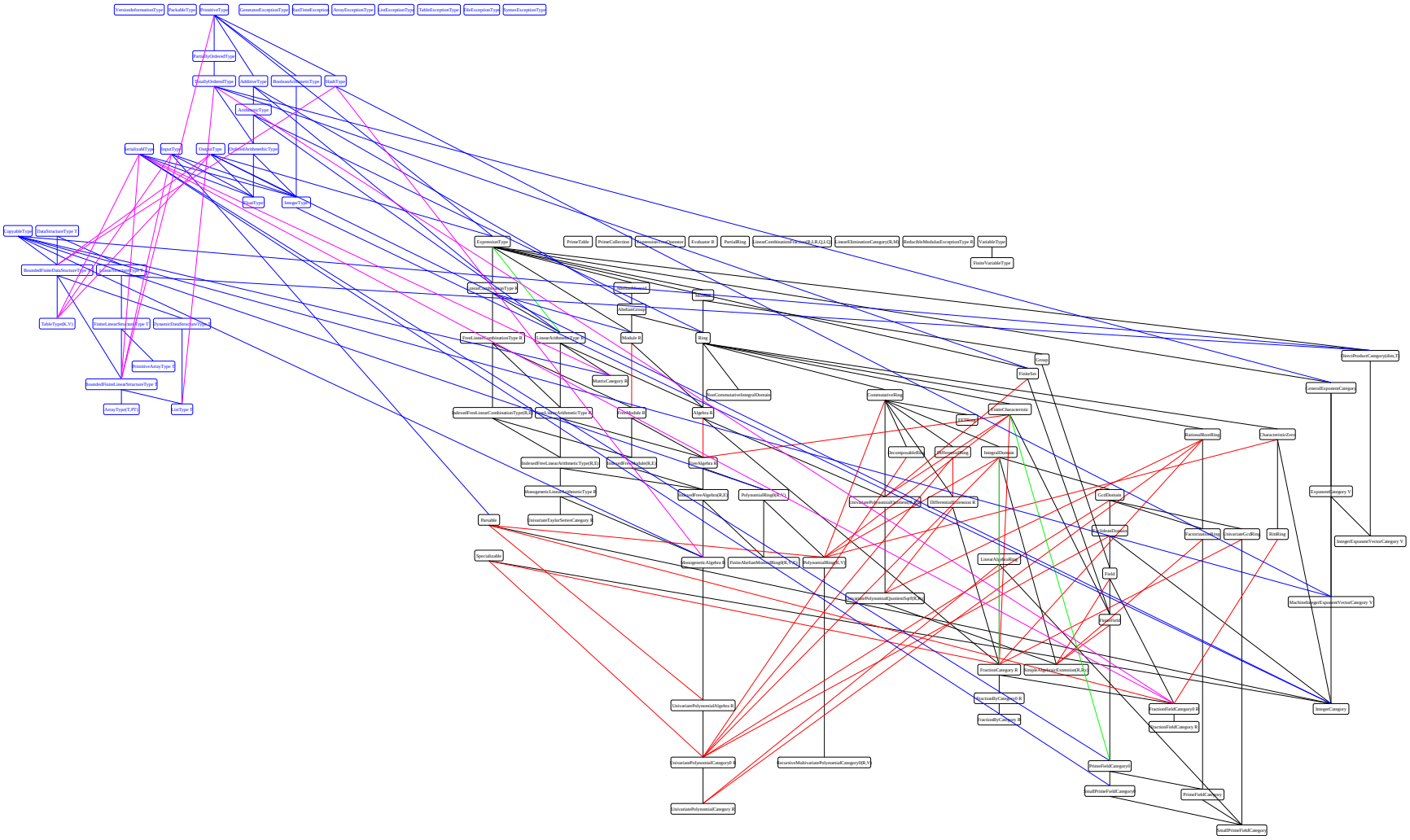
(4) $(R, p) \longmapsto R/pR$

(5) $R \longmapsto \mathrm{Fr}(R)$,

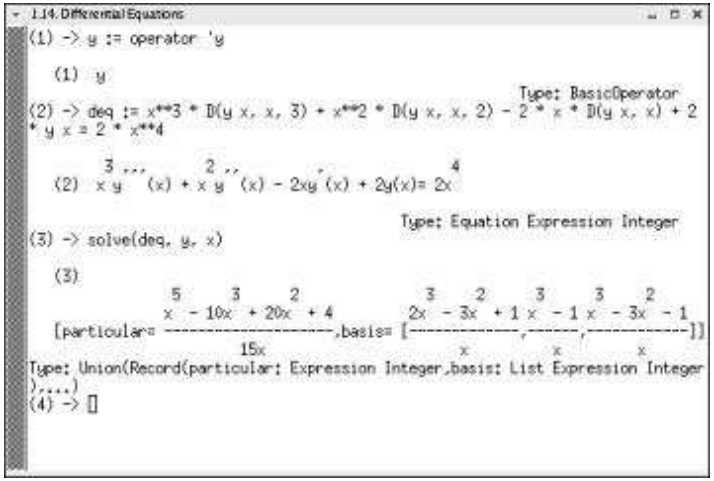
(6) $(R_1, R_2) \longmapsto R_1 \times R_2$.

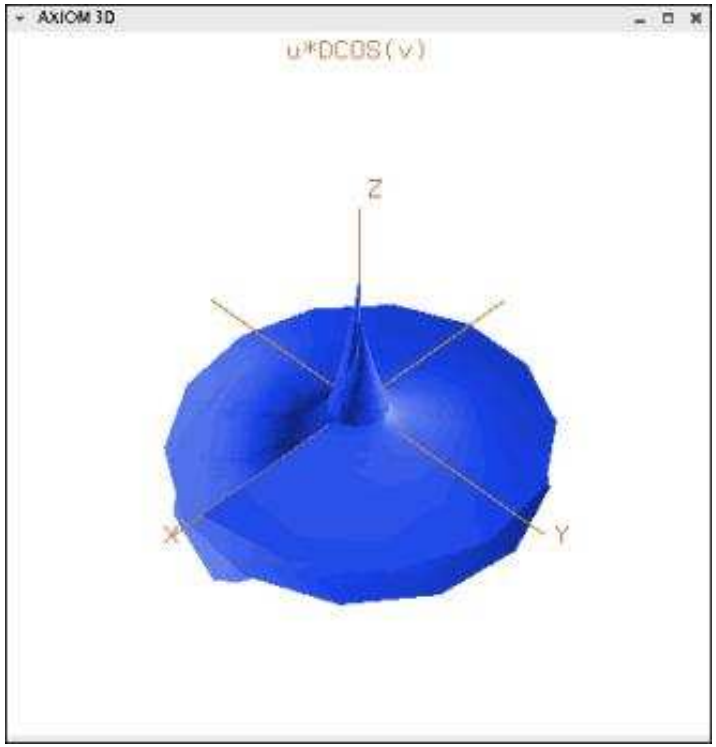
Including conditional statements like:

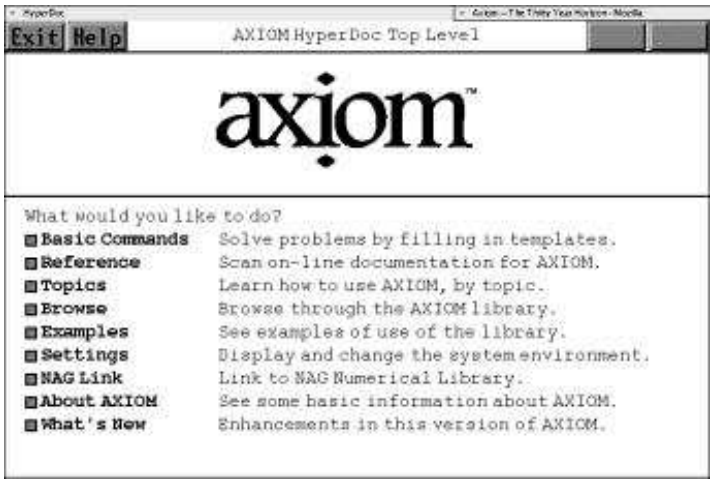
- If R is commutative, then $R[X]$ is commutative.
- If R is an integral domain, then $\mathbf{Fr}(R)$ is a field.

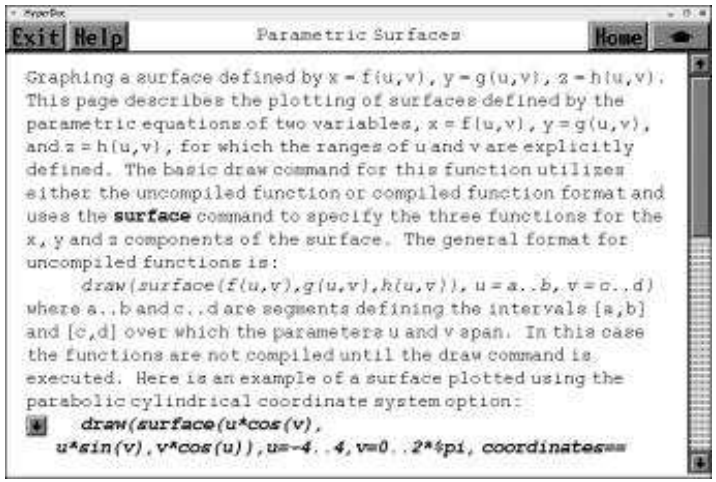


AXIOM: **A laboratory for the mathematician**

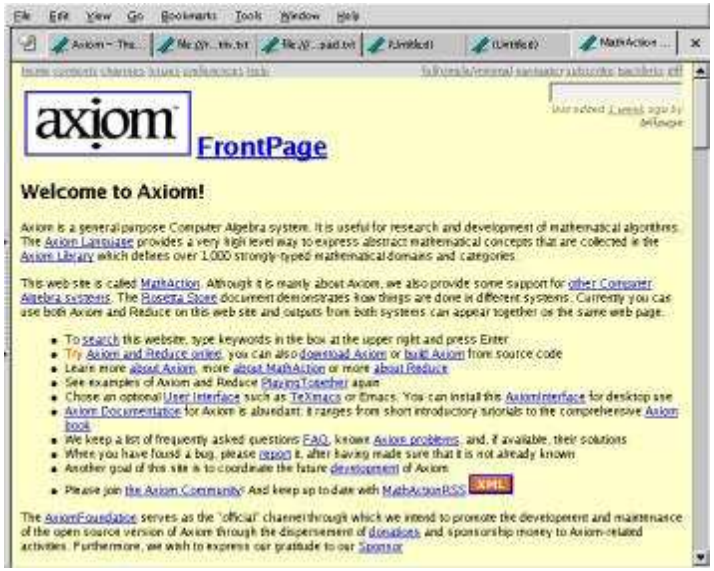


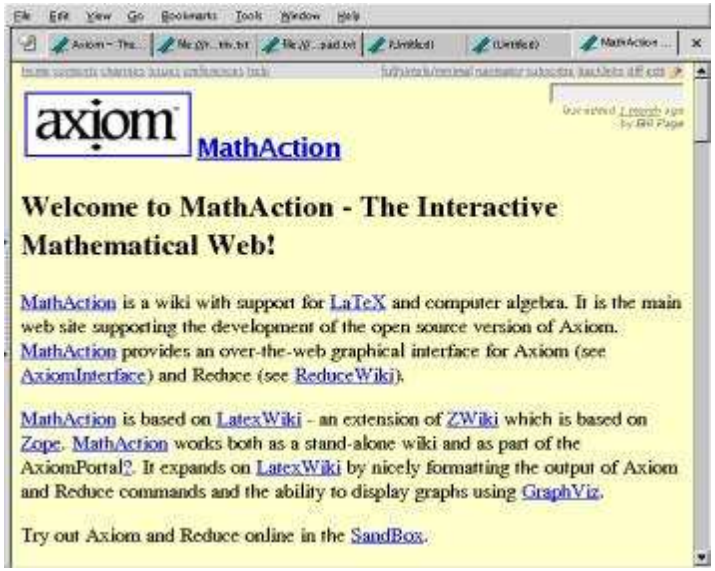


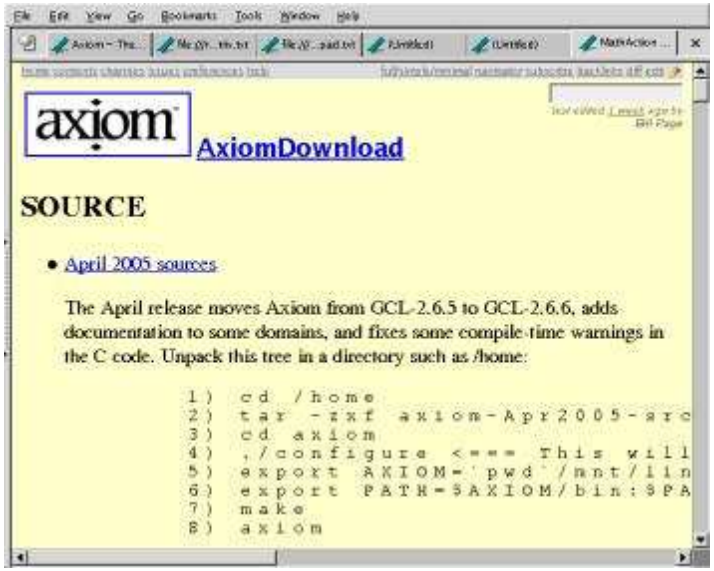




AXIOM: **Free and open source**



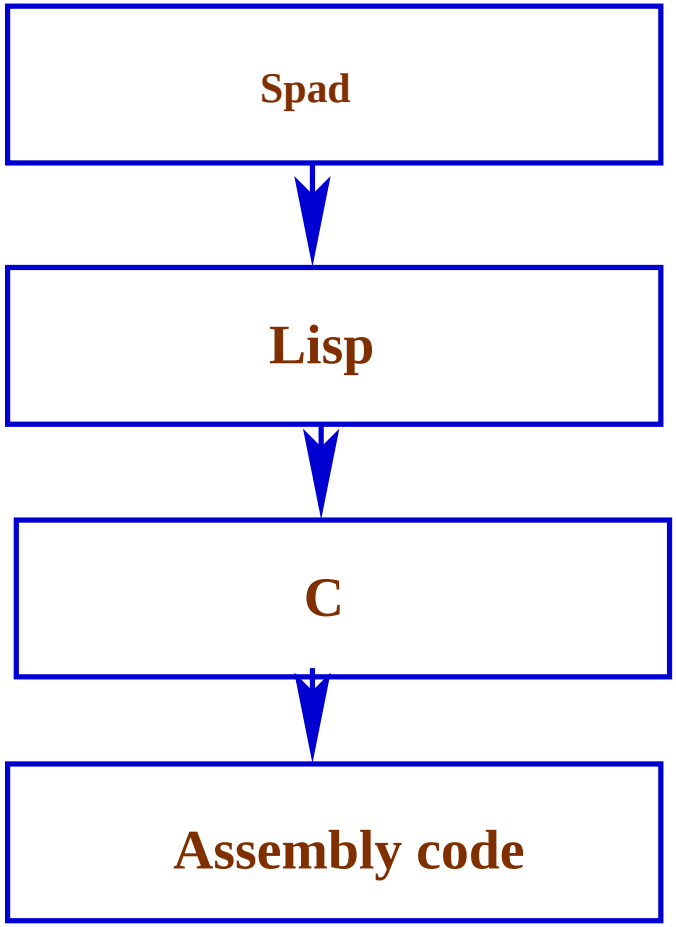




Literate programming

- A literate program should combine the explanation of the code with the actual code.
- **Programming becomes a task of writing for other people rather than the machine.**
- Literate programming was made famous by Knuth who originally created Web for `Pascal` programs.
- Norman Ramsey (Harvard) created noweb which is language independent.
- Axiom uses noweb to make **every file into a literate program.**
- As a result all files in Axiom are meant to be document.
- weave is a command
  - to extract the latex source code,
  - which can help preparing a technical report.

AXIOM: a multiple-level programming environment

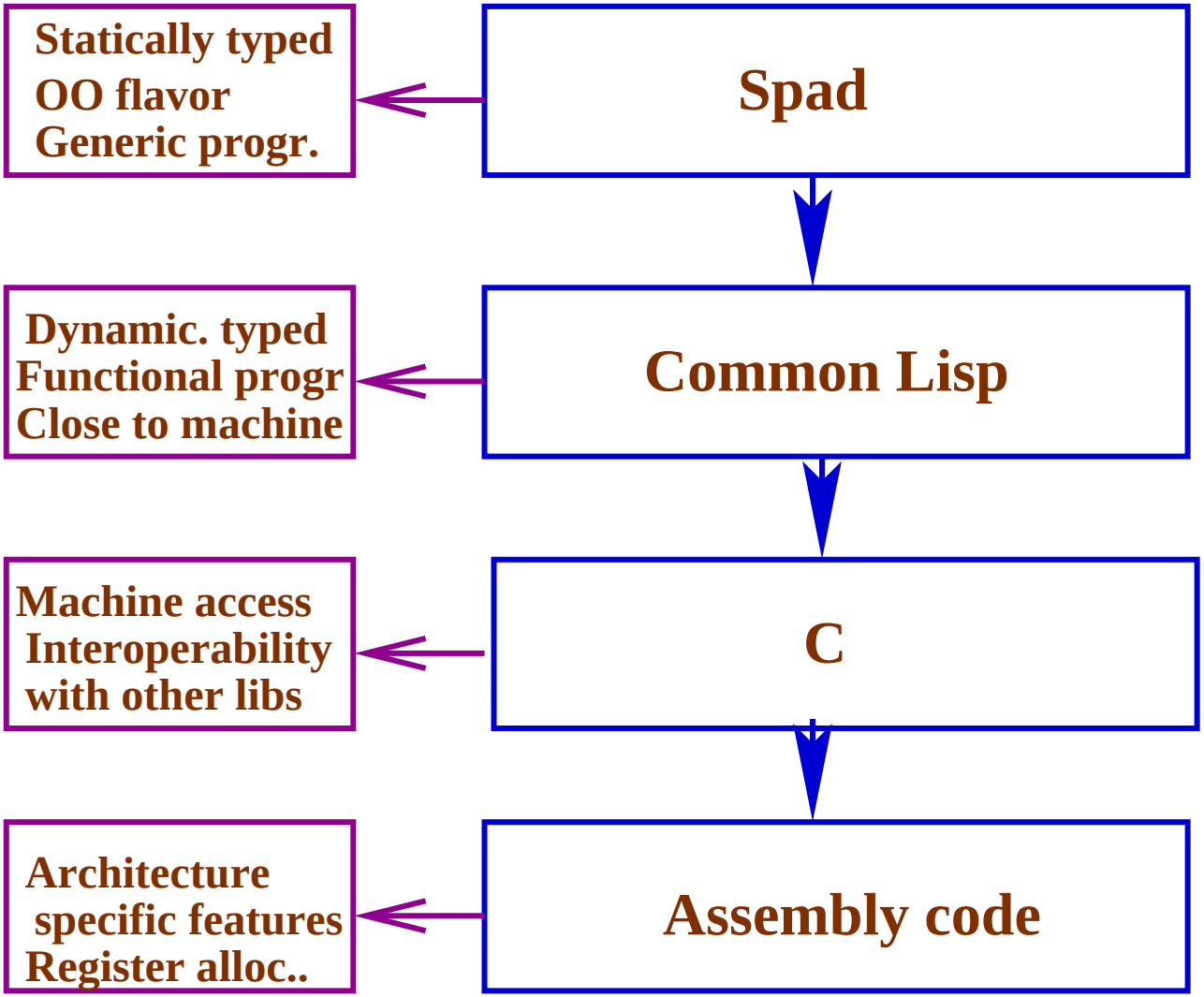


The SPAD language

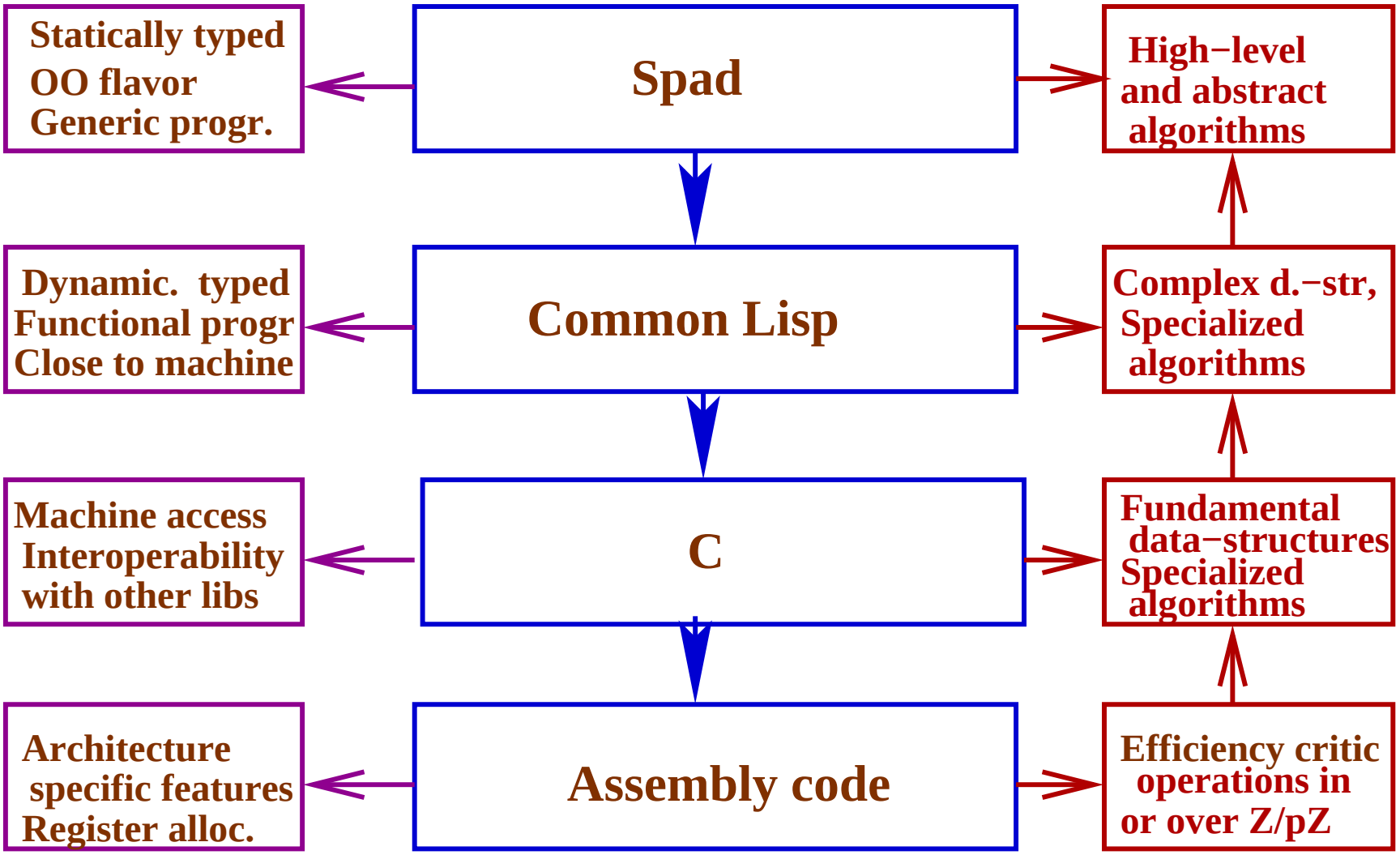
```
ResidueRing(F,Expon,VarSet,FPol,LFPol) : Dom  == Body
 where
   F       :  Field
   Expon   :  OrderedAbelianMonoidSup
   VarSet  :  OrderedSet
   FPol    :  PolynomialCategory(F, Expon, VarSet)
   LFPol   :  List FPol
   Dom   == Join(CommutativeRing, Algebra F) with
     reduce   :   FPol -> $
     ++ reduce(f) produces the equivalence class of f in the res
     coerce   :   FPol -> $
     ++ coerce(f) produces the equivalence class of f in the res
     lift     :     $  -> FPol
     ++ lift(x) return the canonical representative of the equiv
   Body  ==  add
    --representation
      Rep:= FPol
      import GroebnerPackage(F,Expon,VarSet,FPol)
```

```
  relations:= groebner(LFPol)
  relations = [1] => error "the residue ring is the zero rin
--declarations
  x,y: $
--definitions
  0 == 0$Rep
  1 == 1$Rep
  reduce(f : FPol) : $ == normalForm(f,relations)
  coerce(f : FPol) : $ == normalForm(f,relations)
  lift x  == x :: Rep :: FPol
  x + y == x +$Rep y
  -x == -$Rep x
  x*y == normalForm(lift(x *$Rep y),relations)
  (n : Integer) * x == n *$Rep x
  (a : F) * x == a *$Rep x
  x = y == x =$Rep y
  characteristic()      == characteristic()$F
  coerce(x) : OutputForm == coerce(x)$Rep
```

A multiple-level programming environment (I)



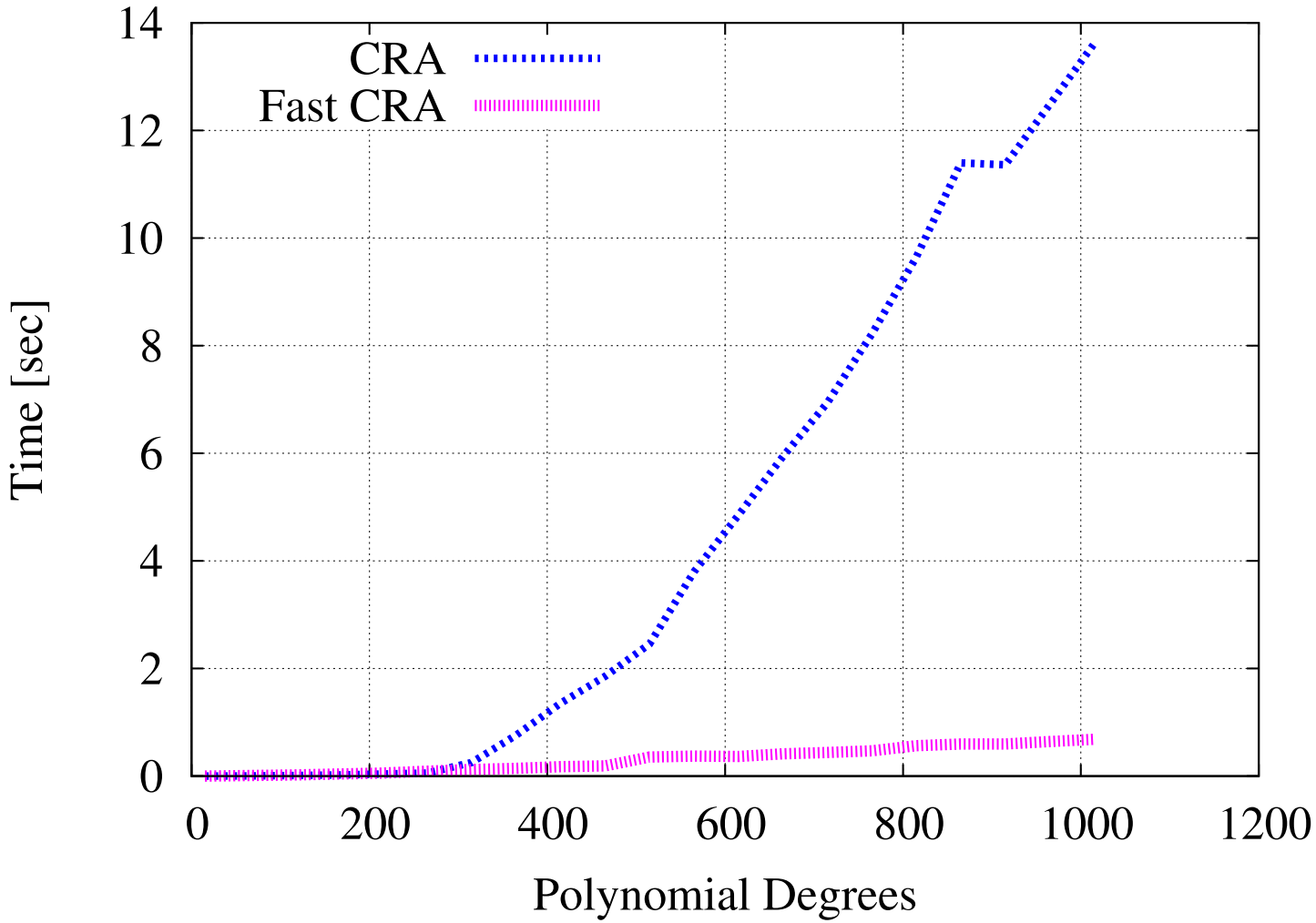
A multiple-level programming environment (II)



A few words on ALDOR

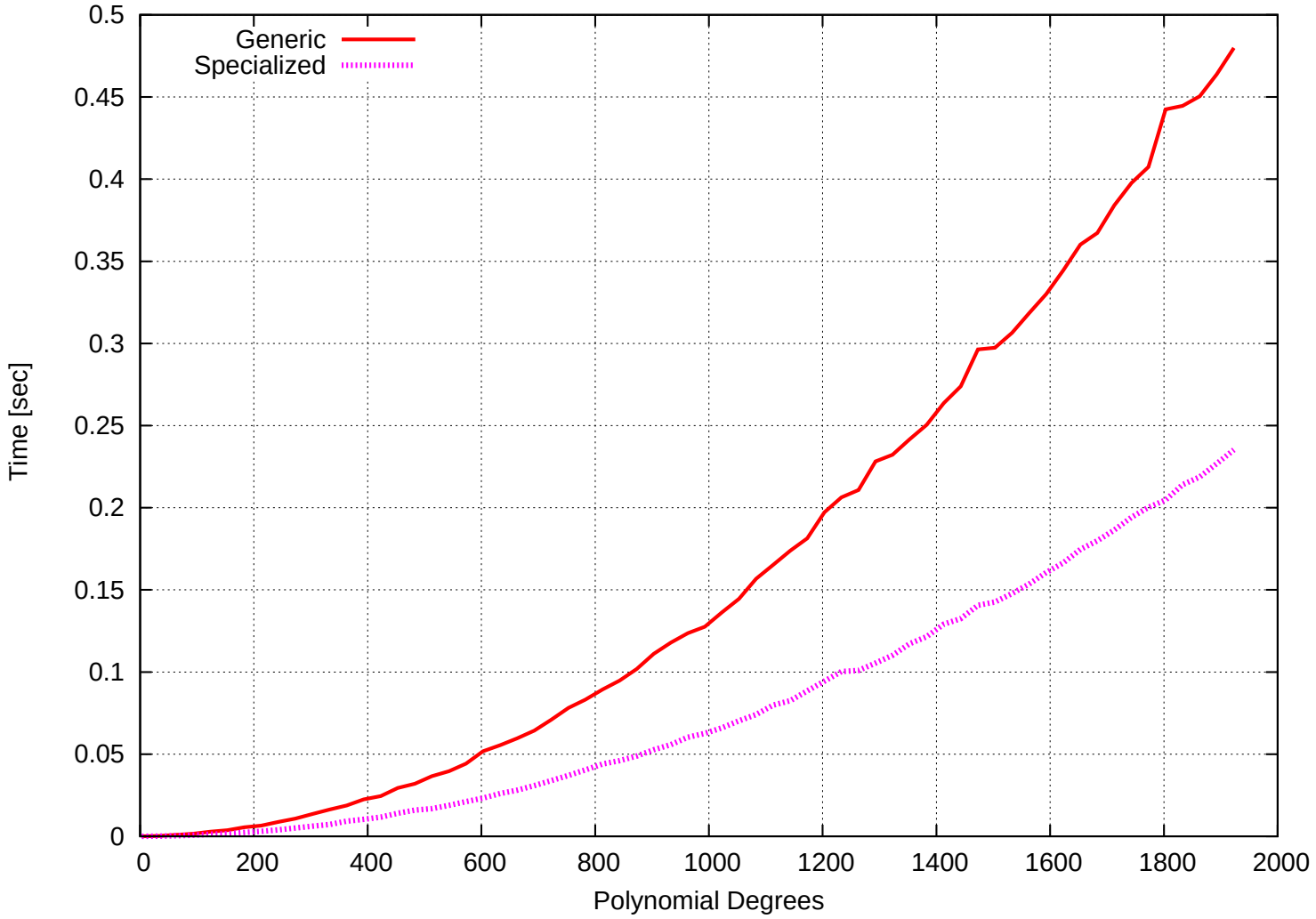
- Some expertise is required in order to interface AXIOM with external packages.
- Developing stand-alone binaries (e.g. servers in a distributed/parallel application) is not purpose of AXIOM.
- ALDOR was initially developed to extend AXIOM via LISP and solve these difficulties.
- ALDOR has also its own libraries:
  - `aldorlib` and `algebra` general purpose libraries for computer algebra,
  - `sumît` for manipulating and solving linear ordinary differential and difference equations and systems
  - `BasicMath` initiated at `NAG` in the 1990's and developed today at `ORCCA`; `BasicMath` supports a **parallel symbolic polynomial system solver**
  - `Calix` for involutive bases in commutative and non-commutative polynomial rings.
  - etc.

Fast vs standard arithmetic



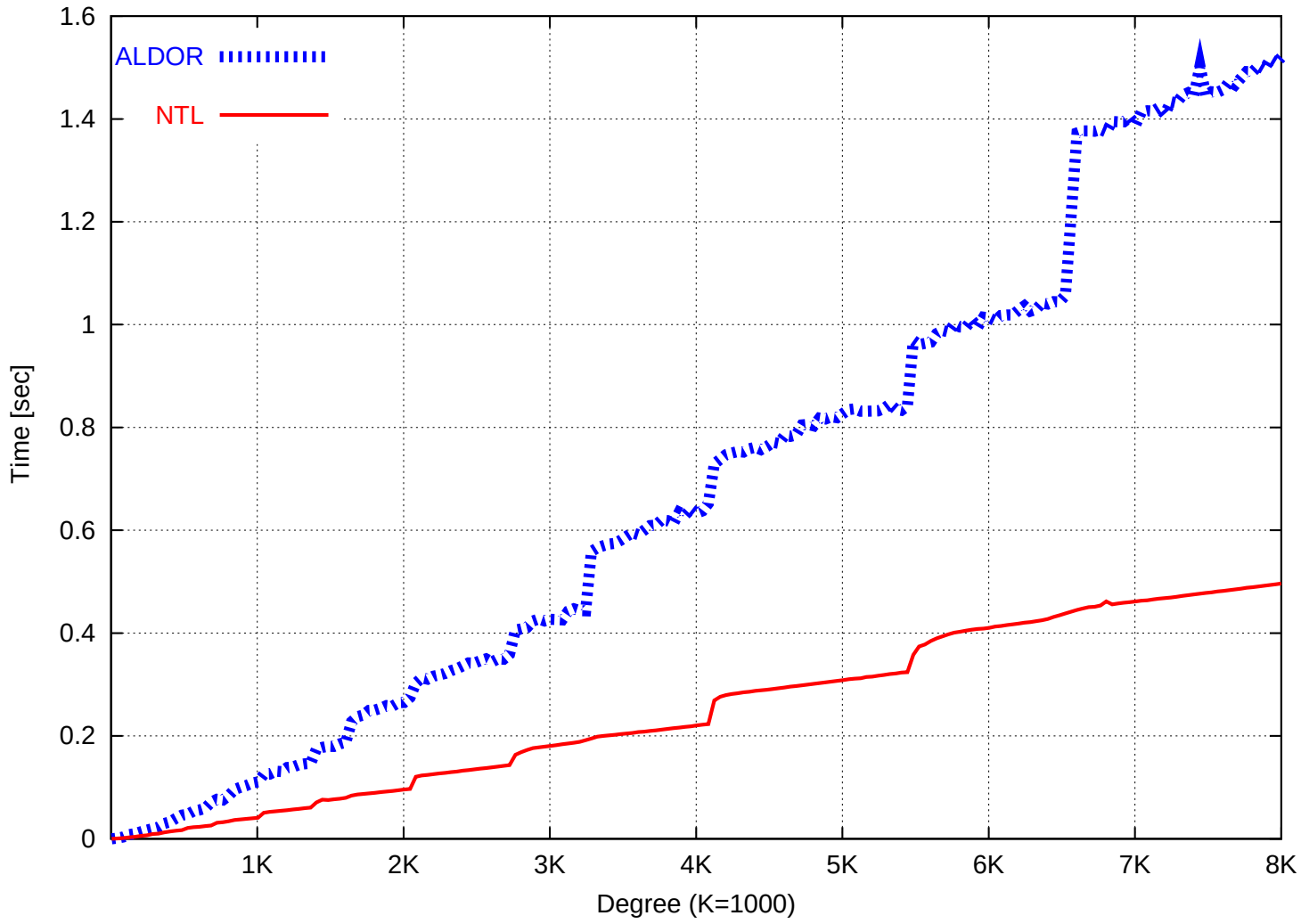
CRA for univariate polynomials in $Z/pZ[x]$ for p a 27-bit prime in ALDOR.

We care about a factor of 2



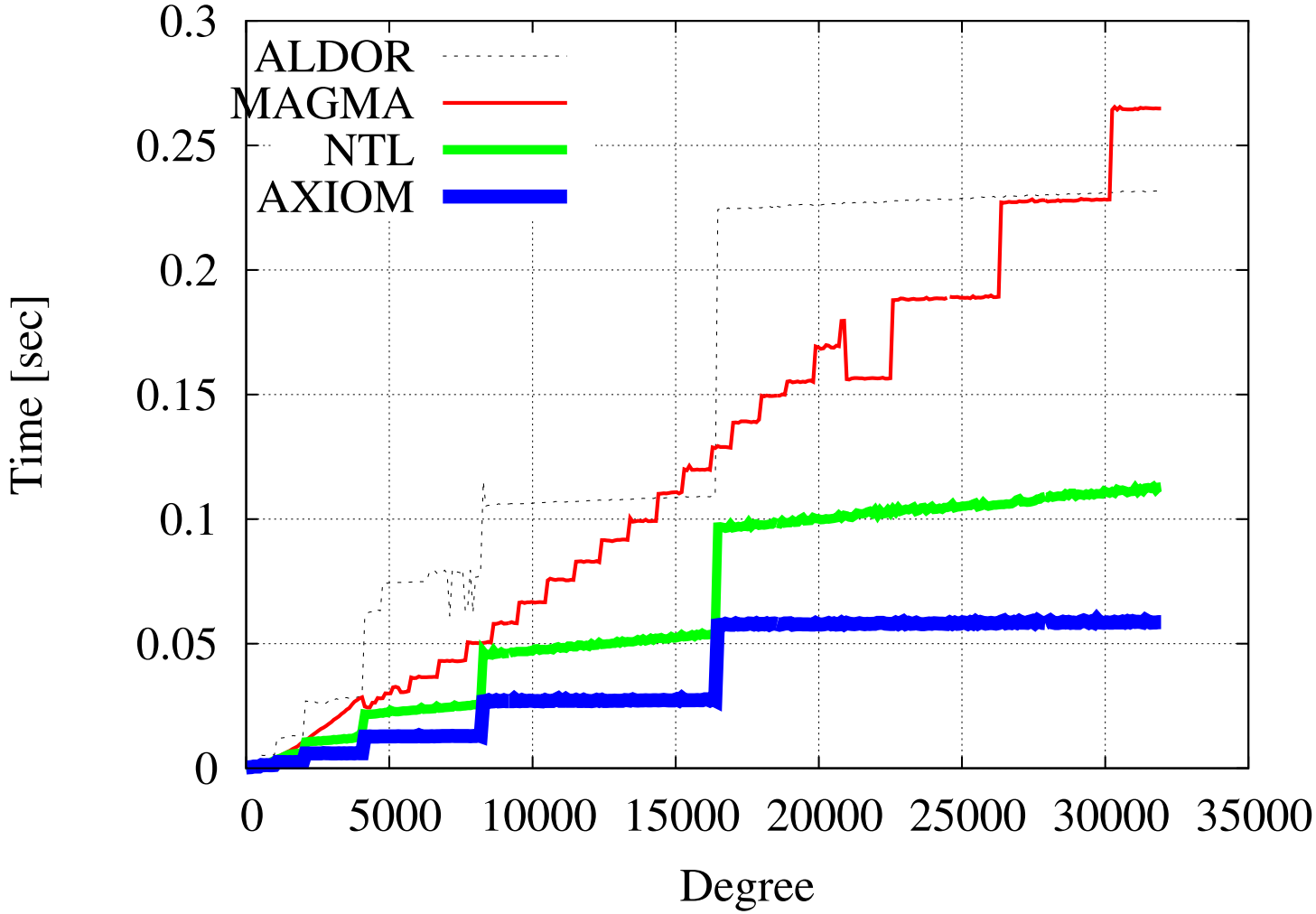
Standard EEA in $Z/pZ[x]$ for a 27-bit prime: generic vs specialized code in ALDOR.

C-level vs ALDOR implementations



Fast EEA in $Z/pZ[x]$ for p a 27-bit prime: generic ALDOR vs specialized C code.

With AXIOM



FFT-based multiplication in $Z/pZ[x]$ for a 27-bit prime.

Conclusions and future work

• The AXIOM/ALDOR language remains a beautiful tool for implementing abstract and generic mathematical algorithms.

• An active community of people has turned AXIOM into a free and open computer algebra system.

• AXIOM and ALDOR appear today as platforms for high-performance computing.

Bibliography

- AXIOM http://page.axiom-developer.org
- ALDOR http://www.aldor.org
- GMP http://swox.com/gmp/
- GNU Common Lisp http://www.gnu.org/software/gcl
- MAGMA http://magma.maths.usyd.edu.au/magma/
- NTL http://www.shoup.net/ntl