

Open Source Computer Algebra Systems

Axiom

David Joyner*

June 4, 2008

This survey will look at Axiom, a free and very powerful computer algebra system available. It is a general purpose CAS useful for symbolic computation, research, and the development of new mathematical algorithms. Axiom is similar in some ways to Maxima, covered in the survey [J1], but different in many ways as well. Axiom, Maxima, and SAGE [S], are the largest of the general-purpose open-source CAS's. If you want to “take a test drive,” Axiom can be tested without installation via the web interface [AS] or the SAGE online interface [S].

1 Axiom's history and language

In the last few years, Axiom has forked *twice*. The original Axiom is headed by Tim Daly. The OpenAxiom fork [O] is headed by Gabriel Dos Rios. The FriCAS fork [F] is headed by Waldek Hebisch.

Axiom's latest release is May 2008 (Axiom uses Month-Year format for release numbers; there are no 'point release' numbers). OpenAxiom's latest release is 1.1.0. FriCAS's latest release is 1.0.2.

In this brief survey, we occasionally abuse terminology by saying “Axiom” when we really mean “Axiom, Fricas, or OpenAxiom”.

1.1 History

Axiom has an interesting history. Axiom began as Scratchpad II, which was developed at IBM in Yorktown Heights, New York back in the early 1980's, under the direction of Richard Jenks¹. The team of developers then were Barry Trager, Robert Sutor, and Stephen Watt; Tim Daly joined the team in the late 1980's. In the 1990's, Scratchpad II was sold to the NAG (Numerical Algorithms Group) in England and renamed to Axiom. In 2001, NAG withdrew Axiom from the market, agreeing in 2002 to release it as free and open source software, distributed under a modified BSD license². This is primarily thanks to the efforts of Tim Daly and Mike Dewar of NAG. Actually, about a year before NAG released Axiom to Tim Daly as open source, they “released” Aldor³ to a group alдор.org headed by Stephen Watt (the original IBM SPAD and Aldor lead developer, now at the University of Western Ontario)⁴. Recently, Aldor was released under a non-open source license which allows for modifications and redistributions for non-commercial purposes.

*Address: Mathematics Department US Naval Academy, Annapolis, MD 21402, USA, wdjoyner@gmail.com

¹Starting in the 1970's, Scratchpad (the first version) was also directed by Richard Jenks. However, it had no connection with Scratchpad II - the name was merely re-used to simplify the IBM “red tape”.

²Such a license is open source, in the sense of OSI <http://www.opensource.org/>, and GPL compatible, in the sense of <http://www.gnu.org/licenses/license-list.html#SoftwareLicenses>.

³Early names for Aldor were A# (at IBM) and Axiom-XL (at NAG). NAG created the legal trade name “Aldor”.

⁴By “released”, I mean that they allowed alдор.org to distribute the Aldor compiler in a certain mixture of source code and binaries.

1.2 Active developers

In addition to Tim Daly, Axiom has several very active, very smart developers. The core active Axiom developers in alphabetical order are Christian Aistleitner (for Aldor), Tim Daly, Gabriel Dos Reis (now developing OpenAxiom), Waldek Hebisch (now developing FriCAS), Ralf Hemmecke, Bill Page, Martin Rubey, William Sit, Grégory Vanuxem, and Stephen Watt (for Aldor). This is leaving out a very long list of contributors and developers (such a list would be pages long and include a who's who of CAS pioneers) but these names appear to be the most active on the developers email list (both for Axiom and for Aldor) in the past 6 months.

1.3 Language

Unless otherwise stated, "Axiom" shall refer to one of these three versions of Axiom.

Apart from a scripting language, Axiom has, in some sense, two extension languages, both of which are strongly and statically typed⁵. The first one, SPAD, is processed by the Axiom compiler (which translates it in to Lisp - GCL to be precise), and is free and open source. The second one is Aldor. A striking difference with most other CAS's, as was emphasized by Martin Rubey, is that Aldor has a very well-defined semantics (see the Aldor compiler users guide, <http://www.aldor.org/docs/aldorug.pdf>). Aldor is available from [A], but unlike Axiom, the license Aldor is released under is not actually open source in the sense of the Open Source Initiative [OSI].

The original motivation for Aldor was to replace SPAD as extension language of the Axiom computer algebra system. The Aldor compiler can be used to generate code which runs within the Axiom system, separately, or linked into other applications. When used as the library compiler for Axiom, Aldor (like SPAD) will translate to GCL, which is then compiled to produce object code. When used as a stand alone compiler, Aldor generates C code directly (which is compiled by a C compiler).

One key point that sets Axiom apart from Maxima, Reduce, Mathematica and Maple is the design of the Axiom mathematical library. The SPAD language (and Aldor) was designed specifically to make it possible to write mathematical algorithms in a flexible but type-correct manner which today might be referred to as a polymorphic object-oriented paradigm, such as you might find in Haskell and to a lesser extent in Python and similar languages. The single most important aspect of the Axiom language is that it has types, and types are (in SPAD: nearly, in Aldor: really) first class objects. Maybe the best way to illustrate their use is to consider the constructor `Fraction Integer`. Take any integral domain, for example, `Integer`. Then the constructed type `Fraction Integer` is the rationals. As another example, if you start with `Polynomial Integer`, then `Fraction Polynomial Integer` is the field of rational functions. These properties of SPAD and/or Aldor are attractive to mathematicians and theoretical computer scientists investigating implementation techniques for various mathematical algorithms (see both [LM] and the article [R], for example). The programmer can write a generic program in a mathematical style, for, say, any integral domain, without having to bother distinguishing between integers and polynomials.

Example 1. *This example, to illustrate Aldor syntax, was given by Bill Page at a recent talk at the University of Washington⁶.*

A prime number sieve to count primes $\leq n$:

```
#pile
#include "axiom.as"

import from Boolean, Integer, NonNegativeInteger
```

⁵I realize different people mean different things by these terms! Please see [PT] for details.

⁶Bill Page, "Introduction to the Axiom Library compiler for Python Programmers", SageDays2 talk at <http://www.sagemath.org:9001/AxiomCompiler> and http://sage.math.washington.edu/sage_days2_audio/.

```
sieve(n: Integer): Integer ==
  isprime: OneDimensionalArray Boolean := new(n::NonNegativeInteger, true)
  np:Integer := 0
  for p in 2..n | isprime p repeat
    np := np + 1
    for i in (p+p)..n by p repeat isprime i := false
  np
```

Axiom also contains an interpreter (or scripting language), implementing a subset of SPAD, by which the user interacts with the Axiom system on the command line.

2 Documentation and capabilities

2.1 Basics

A great deal of information about Axiom is available on its website. Another useful aspect of the Axiom website is the excellent webpage providing a comparison between CAS's:

<http://axiom.axiom-developer.org/axiom-website/rosetta.pdf>

<http://axiom-wiki.newsynthesis.org/RosettaStone>

- Internet:
 - Website :*
 - Fricas: <http://fricas.sourceforge.net/>,
 - OpenAxiom: <http://www.open-axiom.org/index.html>,
 - Axiom: <http://axiom-developer.org/>.
 - General: <http://axiom-wiki.newsynthesis.org/FrontPage> (maintained mostly by Bill Page).
- Documentation:
 - Online reference manual:*
 - <http://axiom.axiom-developer.org/axiom-website/documentation.html>
 - There is also a published Axiom book [JS] (see also [Ax1], [Ax2]). It is also available online, with tutorials and other documentation. Most are in English but there is a French introduction available as a pdf on the website.
 - Bibliography* (originally collected by Nelson Beebe):
 - <http://www.math.utah.edu/pub/tex/bib/axiom.html>.
 - Aldor Compiler User Guide:* <http://www.alдор.org/docs/aldorug.pdf>
- Interfaces:
 - Command line*
 - Emacs mode for Axiom:* Cliff Yapp, <http://axiom-wiki.newsynthesis.org/AxiomEmacsMode>
 - Emacs mode for Aldor:* Ralf Hemmecke,
 - <http://www.risc.uni-linz.ac.at/people/hemmecke/aldor/emacs.html>,
 - <http://www.risc.uni-linz.ac.at/people/hemmecke/aldor/aldor-mode.pdf>
 - Termacs:* <http://axiom-wiki.newsynthesis.org/TeXmacs>
 - Web interface:* The Axiom Sandbox: <http://axiom-wiki.newsynthesis.org/SandBox>
 - The SAGE online interface: <http://www.sagenb.org/>.
- Availability:
 - Source code:*
 - OpenAxiom: <http://www.open-axiom.org/download.html>,

Axiom: <http://github.com/axiom/tree/master>,

Fricas: <http://fricas.sourceforge.net/download.html>.

Binary: <http://wiki.axiom-developer.org/axiom-website/download.html>

An alternative way: Another way to install Axiom, first install SAGE [S], version 3.0 or later, and then install the optional package `fricas-1.0.2`, created by Bill Page and Burcin Erocal. Once the Fricas package is installed, you can use Axiom within SAGE or, running the `axiom` script, in stand-alone mode.

- *Support:*

There is are several email lists for support and development issues, which you will find on the following pages.

Axiom: <http://wiki.axiom-developer.org/axiom-website/community.html>,

Fricas: <http://groups.google.com/group/fricas-devel?hl=en>,

OpenAxiom: <http://www.open-axiom.org/lists.html>.

- *License:*

Axiom is distributed under terms of the Modified BSD license. Axiom was released under this license as of September 3, 2002. As mentioned, Aldor is not open source, though the source code is publicly available from the Aldor webpage.

2.2 Capabilities

Axiom has the ability to manipulate symbolic numerical expressions, including differentiation, integration, Taylor series, Laplace transforms, ordinary differential equations, systems of linear equations, polynomials, and sets, lists, vectors, matrices, and tensors. It has a package for algebraic-geometric codes and function fields over finite fields, a package for Homology computations, and a package for computing with Lyndon words, to name a few. For a list of others, please see the integrated help facility HyperDoc and <http://axiom-wiki.newsynthesis.org/AxiomContributions>.

The Axiom library has an extensive collection of routines designed to implement algebraic routines for diverse areas of mathematics. The Axiom library implements a hierarchy of abstract data structure types, from simple Aggregates to Keyed Dictionaries and Lazy Streams, which collectively make SPAD a very high level programming language. This is important for the representation of mathematical objects and the Axiom library uses these to provide an even more extensive hierarchy of algebraic categories, from Sets to FiniteFields and Partial Differential Rings, and so on. Axiom implements a very wide range of mathematical concepts in the library (over 1300 by one count). Moreover, for many of these mathematical objects, Scratchpad or Axiom was the first CAS where they were implemented (see for example, the book of Davenport [D], which describes one pioneering effort).

Here are a few simple examples: differentiation such as $\frac{d(x^x)}{dx}$, limits such as $\sum_{j=1}^{\infty} \frac{j}{2^j}$, π to 100 digits, power series, symbolic summations, and symbolic matrices.

```
(1) -> D(x^x,x)
```

```
      x      x - 1
(1) log(x)x  + x x
```

Type: Expression Integer

```
(2) -> limit(sum(j/2^j,j=1..n),n=%plusInfinity)
```

```
(2) 2
```

Type: Union(OrderedCompletion Expression Integer,...)

```
(3) -> limit(sum(1/3^j,j=0..n),n=%plusInfinity)
```

```

(3) -
    3
    2
Type: Union(OrderedCompletion Expression Integer,...)
(4) -> digits(100)
(4) 20
Type: PositiveInteger
(5) -> %pi :: Float
(5)
3.1415926535 8979323846 2643383279 5028841971 6939937510 5820974944 592307816
4 0628620899 8628034825 342117068
Type: Float
(6) -> series(x**x, x = 1)
(6)
      2 1      3 1      4 1      5
1 + (x - 1) + (x - 1) + - (x - 1) + - (x - 1) + -- (x - 1)
      2      3      12
+
  3      6 1      7 59      8 71      9
-- (x - 1) - --- (x - 1) + ---- (x - 1) - ---- (x - 1)
40      120      2520      5040
+
  131      10      11
----- (x - 1) + 0((x - 1) )
10080
Type: UnivariatePuisseuxSeries(Expression Integer,x,1)
(7) -> sum(cos((2*r-1)*%pi/(2*n+1)), r=0..5)
      9%pi      7%pi      5%pi      3%pi      %pi
(7) cos(-----) + cos(-----) + cos(-----) + cos(-----) + 2cos(-----)
      2n + 1      2n + 1      2n + 1      2n + 1      2n + 1
Type: Expression Integer
(8) -> xx:= matrix([[a, b], [c, d]])
      +a  b+
(8)  |  |
      +c  d+
Type: Matrix Polynomial Integer
(9) -> determinant xx
(9) a d - b c
Type: Polynomial Integer

```

Many CAS's can do what is given above. To show the advantages of generic programming using types, consider the following matrix of complete symmetric functions, written in the basis of power sum symmetric functions:

```

(10) -> m := matrix [[complete 1, complete 0],[complete 2, complete 1]]
      +      (1)      [] +
      |      |      |
(10) |1      1 2      |
      |- (2) + - (1) (1)|
      +2      2      +
Type: Matrix SymmetricPolynomial Fraction Integer

```

(11) -> determinant m

$$(11) \quad -\frac{1}{2} (2) + \frac{1}{2} (1)$$

Type: SymmetricPolynomial Fraction Integer

Note that Axiom uses the product in the ring of symmetric functions to compute the determinant. To check, by Jacobi-Trudi the result should coincide with the Schur function corresponding to the partition (1, 1):

(12) -> SFunction [1,1]

$$(12) \quad -\frac{1}{2} (2) + \frac{1}{2} (1)$$

Type: SymmetricPolynomial Fraction Integer

Here are some ODEs that Axiom can solve:

(13) -> y:= operator('y);

Type: BasicOperator

(14) -> solve(D(y(x), x) + y(x) * sin x/cos x - 1/cos x = 0, y, x)

(14) [particular= sin(x),basis= [cos(x)]]

Type: Union(Record(particular: Expression Integer,basis: List Expression Integer),...)

(15) -> solve(D(y(x), x, 2)+4*D(y(x), x)+4*y(x)-x*exp(-2*x) = 0, y, x)

$$(15) \quad [\text{particular} = \frac{x^3 - 2x}{6}, \text{basis} = [e^{-2x}, x e^{-2x}]]$$

Type: Union(Record(particular: Expression Integer,basis: List Expression Integer),...)

(16) -> solve(x*(1-x**2)*D(y(x), x) + (2*x**2 - 1)*y(x) - x**3*y(x)**3 = 0, y, x)

$$(16) \quad \frac{(-2x^5 + 4x^2)y(x)^2 + 5x^4 - 5x^2}{5y(x)^2}$$

Type: Union(Expression Integer,...)

Indeed, $y = \sin(x)$ does solve the first order linear ODE $y' + \tan(x)y = \sec(x)$, as Axiom tells us in (14). Secondly, Axiom has no problem with the nasty, but linear second order, constant-coefficient ODE $y'' + 4y' + 4y = xe^{-2x}$ (see (15)). But the last one, Bernoulli's non-linear equation $x(1 - x^2)y' + (2x^2 - 1)y - x^3y^3 = 0$, is much harder to solve⁷. The fact that Axiom had no problem doing this (see (16)) is a good illustration of its unique computational firepower.

⁷I was not able to solve Bernoulli's equation in Maxima, but according to [L97], Macsyma can solve this. In fact, [L97] indicates that Macsyma can solve Bernoulli's equation explicitly, whereas Axiom's solution is implicit. Both types of solutions have their advantages. (Recently, David Billingham has recently written an Axiom package `contrib.ode` which can solve the Bernoulli ODE *explicitly*.)

Martin Rubey [R] has written a package **GUESS** which guesses formulas for a sequence given its first few terms. He was kind enough to send me some examples for this column:

```
(17) -> guessPRec [1,2,3,5,8,13]
```

```
(17)
```

```
[[function= [f(n): f(n + 2) - f(n + 1) - f(n)= 0,f(0)= 1,f(1)= 2],order= 0]]
```

```
(18) -> l := [1, 1, q+1, q**3+q*q+2*q+1, q**6+q**5+2*q**4+3*q**3+3*q*q+3*q+1, _
    q**10+q**9+2*q**8+3*q**7+5*q**6+5*q**5+7*q**4+7*q**3+6*q*q+4*q+1];
```

```
(19) -> guessADE(q)(l, maxPower==2).1.function
```

```
(19)
```

```
BRACKET
```

```
  n
```

```
[x ]f(x):
```

```
  - x f(x)f(q x) + f(x) - 1= 0,f(0)= 1,f'(0)= 1,f''(0)= 2q + 2
```

```
,
```

```
  ,,,      3      2
  f'(0)= 6q  + 6q  + 12q + 6
```

```
,
```

```
  (iv)      6      5      4      3      2
  f'(0)= 24q  + 24q  + 48q  + 72q  + 72q  + 72q + 24
```

```
(20) -> guess([0,1,3,9,33], [guessRat], [guessSum, guessProduct])
```

```

          s - 1
      n - 1  5
      --+  +-+--+
(20) [[function= > | |  p  + 2,order= 0]]
      --+  | |  4
      s = 0  p = 0
          5    4
```

There is currently no other package that can do this.

I can't end this paper without at least mentioning Axiom's ability to compute AG codes, thanks to the PAFF package⁸ by Gaétan Haché. (The curious reader is referred to the introduction of the CCA article [J2] for a very brief background on the mathematics of AG codes - error-correct block codes which one constructs in a certain way from algebraic curves over finite fields.) Gaétan Haché has written a very impressive array of routines implementing function fields over finite fields in Axiom. Consider the projective curve C over $GF(2)$ given by $X^5 + Y^2Z^3 + YZ^4 = 0$. In [H], Haché shows how to use his package to compute a basis of the Riemann-Roch space $L(D)$ for C , where $D = 10(P_1 + P_2 + P_3)$ and P_1, P_2, P_3 are the three places of degree 1 on C . He uses this curve over $GF(2^4)$ to construct an error-correcting code

⁸Though there is no explicit distribution license for the PAFF package given in its documentation or code, in an email to the author dated 11/30/2006, Gaétan Haché said: "I have no problem to release it under GPL or an even less restrictive license such as the MIT one." Gaétan also points out: "in the examples there is an error: an argument is missing when PAFFFF is called - for example, PAFFFF(arg1, arg2) should be replace by PAFFFF(arg1, arg2, BLQT)".

over $GF(16)$ with parameters $[n, k, d] = [33, 11, 21]$. This is another good illustration of Axiom's unusual abilities.

In summary, Axiom has some impressive packages, an active group of talented developers, and a fascinating history. Axiom serves a very important and very useful rôle for certain very advanced mathematical programmers, due to special properties of its language. To find out more, please visit <http://axiom.axiom-developer.org/> or <http://axiom-wiki.newsynthesis.org/>.

2.3 Thanks

It is a pleasure to thank Bill Page and Waldek Hebisch for their help with this article but also the SAGE package fricas-1.0.2, which was used to compute some of the examples above. I also thank Martin Rubey for detailed criticisms and suggestions, including many of the examples above. Tim Daly and Stephen Watt kindly corrected several inaccuracies in an earlier version. Finally, I thank Emil Volcheck for useful suggestions, and Gaetan Hache for kindly allowing me to quote from some private email correspondence. Of course, only I am responsible for any mistakes. If you have corrections or comments, please email me.

2.4 Axiom workshop, July 24-26, 2008

We end this survey with a brief advertisement for the upcoming Aldor and Axiom Workshop at RISC in Austria, organized by Ralf Hemmecke and Martin Rubey. Conference webpage:

<http://axiom-wiki.newsynthesis.org/WorkShopRISC2008>. From that webpage: "The workshop aims at a cooperation of Aldor, Axiom, OpenAxiom, and FriCAS developers with developers of packages written for other Computer Algebra Systems, and mathematicians that would like to use a computer algebra system to perform experiments."

References

- [A] Aldor website, <http://www.aldor.org/>
- [AS] The Axiom Sandbox, <http://axiom-wiki.newsynthesis.org/SandBox>
- [Ax1] Timothy Daly, et al, **Axiom: The 30 year horizon**, 1100+ page rewrite of [JS], 2003. Available at: <http://portal.axiom-developer.org/public/book2.pdf>.
- [Ax2] Timothy Daly, **Axiom Volume 1: Tutorial**, Lulu.com publishers, Dec 29, 2005 ISBN 141166597X.
- [D] James H. Davenport, **On the Integration of Algebraic Functions**, (Lecture Notes in Computer Science, volume 102), Springer-Verlag, 1982.
- [F] FriCAS, Waldek Hebisch maintainer, <http://fricas.sf.net/>.
- [H] Gaétan Haché, *Example of Axiom package PAFF*. Available at: <http://axiom-wiki.newsynthesis.org/PAFF>
- [JS] Richard D. Jenks and Robert S. Sutor, **Axiom**, Springer-Verlag, 1992. <http://portal.acm.org/citation.cfm?id=131846&dl=>
- [J1] D. Joyner, *OSCAS Maxima*, ACM Communications in Computer Algebra, Vol 40(2006)92-96 <http://www.sigsam.org/cca/issues/issue157.html>
- [J2] D. Joyner, *Conjectural permutation decoding of some AG codes*, ACM Communications in Computer Algebra, Vol 39(2005)26-32. <http://www.sigsam.org/cca/issues/issue151.html>

- [LM] X. Li and M. Moreno Maza, *Efficient implementation of polynomial arithmetic in a multiple-level programming environment* In **Proc. International Congress of Mathematical Software - ICMS 2006**, Springer, 2006, pages 12-23.
- [L97] Richard Liska, Ladislav Drska, Jiří Limpouch, Milan Sinor, Michael Wester, Franz Winkler, *Computer Algebra - algorithms, systems and applications*, June 2, 1997. Available: <http://kfe.fjfi.cvut.cz/~liska/ca/all.html>.
- [O] OpenAxiom, Gabriel Dos Reis maintainer, <http://www.open-axiom.org/index.html>
- [OSI] Open Source Initiative, <http://www.opensource.org/index.html>
- [PT] E. Poll and S. Thompson, *The type system of Aldor*, preprint, 1999. Available at: <http://citeseer.ist.psu.edu/294143.html>
- [R] Martin Rubey, *Extended Rate, more GFUN*, preprint 2007. Available at: <http://front.math.ucdavis.edu/math.CO/0702086>.
- [S] The SAGE Group, *Mathematical Software*, version 3.0.2, <http://www.sagenb.org/> (for SAGE online), and <http://www.sagemath.org/>.

David Joyner
Mathematics Department
US Naval Academy
572C Holloway Road
Annapolis, MD 21402 USA
wdjoyner@gmail.com