axiom™

The 30 Year Horizon

Manuel Bronstein *William Burge* *Timothy Daly*
James Davenport *Michael Dewar* *Martin Dunstan*
Albrecht Fortenbacher *Patrizia Gianni* *Johannes Grabmeier*
Jocelyn Guidry *Richard Jenks* *Larry Lambe*
Michael Monagan *Scott Morrison* *William Sit*
Jonathan Steinbach *Robert Sutor* *Barry Trager*
Stephen Watt *Jim Wen* *Clifton Williamson*

Volume 12: Axiom Crystal

January 22, 2018

6b082ddb9512e0fc4348d4700cd785228393568d

Inclusion of names in the list of credits is based on historical information and is as accurate as possible. Inclusion of names does not in any way imply an endorsement but represents historical influence on Axiom development.

Michael Albaugh
Christian Aistleitner
Jerry Archibald
Henry Baker
Yurij Baransky
Gerald Baumgartner
Nelson H. F. Beebe
Fred Blair
Raoul Bourquin
Wolfgang Brehm
Manuel Bronstein
Florian Bundschuh
Ralph Byers
Robert Cavines
Tzu-Yi Chen
David V. Chudnovsky
James Cloos
Christophe Conil
Robert Corless
Karl Crary
Nathaniel Daly
James H. Davenport
Didier Deshommes
Jack Dongarra
Claire DiCrescendo
Zlatko Drmac
Lee Duhem
Dominique Duval
Heow Eide-Goodman
Richard Fateman
John Fletcher
George Frances
Korrinn Fu
Van de Geijn
Gustavo Goertkin
Teresa Gomez-Diaz
Johannes Grabmeier
James Griesmer
Ming Gu
Steve Hague
Mike Hansen
Bill Hart
Arthur S. Hathaway
Karl Hegbloom
Antoine Hersen
Roger House
Alejandro Jakubi
William Kahan
Grant Keady
David Kincaid

Cyril Alberga
Richard Anderson
S.J. Atkins
Martin Baker
David R. Barton
Gilbert Baumslag
Jay Belanger
Vladimir Bondarenko
Alexandre Bouyer
Peter A. Broadbery
Christopher Brown
Luanne Burns
Quentin Carpent
Bruce Char
Bobby Cheng
Gregory V. Chudnovsky
Jia Zhao Cong
Don Coppersmith
Gary Cornell
Jeremy Du Croz
Timothy Daly Sr.
David Day
Michael Dewar
Jean Della Dora
Sam Dooley
Lionel Ducos
Martin Dunstan
Robert Edwards
Lars Erickson
Bertfried Fauser
Brian Ford
Constantine Frangos
Marc Gaetano
Kathy Gerber
Samantha Goldrich
Laureano Gonzalez-Vega
Matt Grayson
Vladimir Grinberg
Jocelyn Guidry
Satoshi Hamaguchi
Richard Hanson
Vilya Harvey
Dan Hatton
Ralf Hemmecke
Nicholas J. Higham
Gernot Hueber
Richard Jenks
Kyriakos Kalorkoti
Wilfrid Kendall
Keshav Kini

Roy Adler
George Andrews
Jeremy Avigad
Stephen Balzac
Thomas Baruchel
Michael Becker
David Bindel
Mark Botch
Karen Braman
Martin Brock
Stephen Buchwald
William Burge
Pierre Casteran
Ondrej Certik
Cheekai Chin
Mark Clements
Josh Cohen
George Corliss
Meino Cramer
David Cyganski
Timothy Daly Jr.
James Demmel
Inderjit Dhillon
Gabriel Dos Reis
Nicolas James Doye
Iain Duff
Brian Dupee
Hans-Dieter Ehrich
Mark Fahey
Stuart Feldman
Albrecht Fortenbacher
Timothy Freeman
Rudiger Gebauer
Patricia Gianni
Holger Gollan
Stephen Gortler
Klaus Ebbe Grue
Oswald Gschnitzer
Gaetan Hache
Sven Hammarling
Richard Harke
Martin Hassner
Waldek Hebisch
Henderson
Hoon Hong
Pietro Iglio
Bo Kagstrom
Kai Kaminski
Tony Kennedy
Ted Kosan

Paul Kosinski
Klaus Kusche
Larry Lambe
George L. Legendre
Michel Levaud
Ren-Cang Li
Michael Lucks
Francois Maltey
Bob McElrath
Ian Meikle
Gerard Milmeister
Michael Monagan
Joel Moses
Patrice Naudin
Godfrey Nolan
Michael O'Connor
Humberto Ortiz-Zuazaga
David Parnas
Didier Pinchon
Frank Pfenning
Gregorio Quintana-Orti
Andre Platzer
Arthur C. Ralfs
Guilherme Reis
Michael Richardson
Jean Rivlin
Raymond Rogers
Jeff Rutter
Alfred Scheerhorn
Martin Schoenert
Fritz Schwarz
Nick Simicich
Jacob Nyffeler Smith
Jonathan Steinbach
Klaus Sutner
Eugene Surowitz
T. Doug Telford
Balbir Thomas
Francoise Tisseur
Barry Trager
Kresimir Veselic
Stephen Watt
Juergen Weiss
James Wen
R. Clint Whaley
Berhard Will
Shmuel Winograd
Waldemar Wiwianka
Liu Xiaojun
Qian Yun
Evelyn Zoernack
Igor Kozachenko
Bernhard Kutzler
Kaj Laurson
Franz Lehner
Howard Levy
Rudiger Loos
Richard Luczak
Osni Marques
Michael McGettrick
David Mentre
Mohammed Mobarak
Marc Moreno-Maza
Mark Murray
C. Andrew Neff
Arthur Norman
Summat Oemrawsingh
Julian A. Padget
Susan Pelzel
Ayal Pinkus
Jose Alfredo Portes
Beresford Parlett
Peter Poromaas
Norman Ramsey
Huan Ren
Jason Riedy
Nicolas Robidoux
Michael Rothstein
Philip Santas
William Schelter
Marshall Schor
Steven Segletes
William Sit
Matthieu Sozeau
Fabio Stumbo
Robert Sutor
Yong Kiam Tan
James Thatcher
Mike Thomas
Steve Toleque
Themos T. Tsikas
Christof Voemel
Andreas Weber
M. Weller
Thorsten Werther
James T. Wheeler
Clifton J. Williamson
Robert Wisbauer
Knut Wolf
Clifford Yapp
Vadim Zhytnikov
Bruno Zuercher
Fred Krogh
Tim Lahey
Charles Lawson
Frederic Lehobey
J. Lewis
Craig Lucas
Camm Maguire
Alasdair McAndrew
Edi Meier
Victor S. Miller
H. Michael Moeller
Scott Morrison
William Naylor
John Nelder
Jinzhong Niu
Kostas Oikonomou
Bill Page
Michel Petitot
Frederick H. Pitts
E. Quintana-Orti
A. Petitet
Claude Quitte
Anatoly Raportirenko
Albert D. Rich
Renaud Rioboo
Simon Robinson
Martin Rubey
David Saunders
Gerhard Schneider
Frithjof Schulze
V. Sima
Elena Smirnova
Ken Stanley
Christine Sundaresan
Moss E. Sweedler
Max Tegmark
Laurent Thery
Dylan Thurston
Raymond Toy
Gregory Vanuxem
Bernhard Wall
Jaap Weel
Mark Wegman
Michael Wester
John M. Wiley
Stephen Wilson
Sandra Wityak
Yanyang Xiao
David Yun
Richard Zippel
Dan Zwillinger

Contents

New Foreword

On October 1, 2001 Axiom was withdrawn from the market and ended life as a commercial product. On September 3, 2002 Axiom was released under the Modified BSD license, including this document. On August 27, 2003 Axiom was released as free and open source software available for download from the Free Software Foundation's website, Savannah.

Work on Axiom has had the generous support of the Center for Algorithms and Interactive Scientific Computation (CAISS) at City College of New York. Special thanks go to Dr. Gilbert Baumslag for his support of the long term goal.

The online version of this documentation is roughly 1000 pages. In order to make printed versions we've broken it up into three volumes. The first volume is tutorial in nature. The second volume is for programmers. The third volume is reference material. We've also added a fourth volume for developers. All of these changes represent an experiment in print-on-demand delivery of documentation. Time will tell whether the experiment succeeded.

Axiom has been in existence for over thirty years. It is estimated to contain about three hundred man-years of research and has, as of September 3, 2003, 143 people listed in the credits. All of these people have contributed directly or indirectly to making Axiom available. Axiom is being passed to the next generation. I'm looking forward to future milestones.

With that in mind I've introduced the theme of the "30 year horizon". We must invent the tools that support the Computational Mathematician working 30 years from now. How will research be done when every bit of mathematical knowledge is online and instantly available? What happens when we scale Axiom by a factor of 100, giving us 1.1 million domains? How can we integrate theory with code? How will we integrate theorems and proofs of the mathematics with space-time complexity proofs and running code? What visualization tools are needed? How do we support the conceptual structures and semantics of mathematics in effective ways? How do we support results from the sciences? How do we teach the next generation to be effective Computational Mathematicians?

The "30 year horizon" is much nearer than it appears.

Tim Daly
CAISS, City College of New York
November 10, 2003 ((iHy))

Axiom Crystal Design

1.1 Book presentation

In the book "Science at the Edge" by John Brockman (ISBN 978-1-4027-5450-0), in the chapter "The second coming – A manifesto" by David Gelernter, David talks about the way we interact with computers. This has some bearing on the crystal notion.

1.1.1 Book spines

David points out that we currently have a "desktop metaphor" which allows us to view our computer interactions as though we were moving things around on a desktop, typically folders and documents. There are several limitations of this metaphor.

The first is that there is a limited amount of space on the desktop. He proposes the idea of a landscape where the computer is just a moving window. This gives much more real estate to hold information.

The lack of desktop space leads to the icon idea to capture a small representation of a document or folder. There are limitations to how representative such a tiny image can be of the original. A book spine is an excellent representation of the contents of a book but a tiny picture of a folder, not so much.

If I look at this idea in terms of the Crystal concept I can see two parallels. The first idea (desktop/icon) vs (landscape/book) is related to the organization of Axiom. There is an ongoing effort to organize the whole of the system into some small number of books. The whole system is then somewhat similar to an encyclopedia where there is a shelf of related information.

Currently the algebra books are on the order of 5000 pages of raw material. They will likely grow many times that size as literate information is added. One website representation would show the Axiom books as book-spines where the algebra section could be broken up (visually, not actually) as encyclopedia-like images. Thus, you would find the algebra "books" from A-C, D-F, etc.

1.1.2 Linking information

A second idea from the book is the limitations of the hierarchical file system idea. Why does a particular file have to have a name? Why does a particular file only live in one folder?

For the first question, he comments that if you had 3 dogs it is reasonable to name them.

But if you have 10,000 cows it probably is not. Some information can be anonymous.

For the second question, he asks why doesn't a folder "grab" the information so that a particular file might not reside in multiple folders. Unix has this idea embodied in links but Windows doesn't support the idea.

He suggests that it might be reasonable to have the folders be active so that a particular piece of information, say a travel receipt, might be "grabbed" by the taxes folder and the travel expense folder.

Crystal's view of this is somewhat different. Information isn't named. It resides in "the problem" floating in space. The naming of information is related to the view.

So if we take a problem in space, say all of your financial information and wrap a crystal around it we can view it in multiple ways, each of which represents a "facet". Moving between these views corresponds to rotating the crystal to view "the problem" through a different facet.

So, in a financial crystal, you might have a taxes facet, a travel expense facet, an assets facet, a checkbook facet, etc. A travel receipt from a business trip which was added to "the problem' would show up in all of these facets in different ways. It is up to the facet to organize this piece of information into its proper place based on the intent of the facet.

"The problem" just is. The meaning of the problem, the division of the problem into parts, the naming of the parts, the organization of the parts, indeed, the very idea that a problem has parts is a function of the facet, not a function of the problem.

Experiments

1.2 Hide/Show a div element

Here we demonstrate the ability to hide or show a named div element.

— **hide/show a div element** —

```
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/xml" charset="us-ascii"/>
  <style>
   html { color:#000000; }
  </style>
  <script language="JavaScript" type="text/javascript">
   function hideshow(flag) {
     var c = document.getElementById('crystal');
     c.style.display=flag;
   }
  </script>
 </head>
 <body>
it works
  <div id="crystal" style="overflow:hidden;display:none">
   this is visible
  </div>
 </body>
 <hr/>
 <a href="javascript:hideshow('none')">Hide</a>
 <a href="javascript:hideshow('block')">Show</a>
</html>
```

———-

1.3 Hide/Show a nested div element

Now that we can hide or show a div element we demonstrate the ability to hide or show a nested div element.

— **hide/show a nested div element** —

```
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/xml" charset="us-ascii"/>
```

```
  <style>
   html { color:#000000; }
  </style>
  <script language="JavaScript" type="text/javascript">
   function showhide(id,flag) {
     var c = document.getElementById(id);
     c.style.display=flag;
   }
   function toggle(id) {
     var c = document.getElementById(id);
     if (c.style.display == 'block') {
       c.style.display='none'
     } else {
       c.style.display='block'
     }
   }
  </script>
 </head>
 <body>
it works
  <div id="crystal" style="overflow:hidden;display:none">
   <a href="javascript:toggle('facet1','block')">
    integrate(sin x,x)
   </a>
   <div id="facet1" style="overflow:hidden;display:none">
    <a href="javascript:showhide('facet1','none')">
    <br/>
     -cos(x)
    </a>
   </div>
  </div>
 </body>
 <hr/>
 <a href="javascript:showhide('crystal','none')">Hide</a>
 <a href="javascript:showhide('crystal','block')">Show</a>
</html>
```

———-

1.4 Hide/Show a ring of elements

Now that we can hide or show a div element we demonstrate the ability to hide or show a ring of div elements. There are 3 elements in the ring, 'facet1', 'facet2', and 'facet3'. Each facet can open or close the associated 'answer' sub-div element.

— hide/show a ring of elements —

```
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/xml" charset="us-ascii"/>
  <style>
   html { color:#000000; }
  </style>
  <script language="JavaScript" type="text/javascript">
```

```
   var circle = ['facet1','facet2','facet3']
   var here = 'facet1';
   var herept = 0;
   function showhide(id,flag) {
     var c = document.getElementById(id);
     c.style.display=flag;
   }
   function toggle(id) {
     var c = document.getElementById(id);
     if (c.style.display == 'block') {
       c.style.display='none'
     } else {
       c.style.display='block'
     }
   }
   /* hide the old, get the next one in the circle, show it */
   function docircle() {
     var c = document.getElementById(here);
     c.style.display='none'
     if (herept == 2) {
       herept = 0 ;
     } else {
       herept = herept + 1;
     }
     here = circle[herept];
     c = document.getElementById(here);
     c.style.display='block'
   }
  </script>
 </head>
 <body onload="showhide('facet1','block')">
it works
  <div id="facet1" style="overflow:hidden;display:none">
   <a href="javascript:docircle()">
    integrate(cos x,x)
   </a>
   <br/><a href="javascript:toggle('facet1a')">toggle</a>
   <div id="facet1a" style="overflow:hidden;display:none">
    <a href="javascript:showhide('facet1a','none')">
    <br/>
     <pre>
     sin(x)
     </pre>
    </a>
   </div>
  </div>
  <div id="facet2" style="overflow:hidden;display:none">
   <a href="javascript:docircle()">
    integrate(sin x,x)
   </a>
   <br/><a href="javascript:toggle('facet2a')">toggle</a>
   <div id="facet2a" style="overflow:hidden;display:none">
    <a href="javascript:showhide('facet2a','none')">
    <br/>
```

```
     <pre>
     -cos(x)
     </pre>
    </a>
   </div>
  </div>
  <div id="facet3" style="overflow:hidden;display:none">
   <a href="javascript:docircle()">
    integrate(tan x,x)
   </a>
   <br/><a href="javascript:toggle('facet3a')">toggle</a>
   <div id="facet3a" style="overflow:hidden;display:none">
    <a href="javascript:showhide('facet3a','none')">
    <br/>
     <pre>
                 2
       log(tan(x) + 1)
       ---------------
                 2
     </pre>
    </a>
   </div>
  </div>
 </body>
 <hr/>
</html>
```

———-

Other work

1.5 Understanding the Dynamics of Complex Lisp Programs [Loet09]

Abstract: Recent advances in web technologies and the availability of robust Lisp libraries supporting them have made it possible to think of new ways of understanding and debugging large applications. In this paper, we will discuss two basic ideas for assessing and verifying the behavior of Lisp programs. First, we propose to use a web browser for graphically displaying debug output in a similar but more versatile way as the Lisp listener is normally used to print output traces. And second, we propose a method for creating HTML visualisations of complex data and control structures that don't trade in level of detail for readability. We will introduce GTFL (a Graphical Terminal For Lisp), which we have implemented based on these two ideas, and discuss its applications.

This paper is of interest, not for its lisp tracing output, but for its ability to pipeline output to a browser and the technology that underlies the whole of it.

GTFL [Loet00] uses Hunchentoot [Weit06] as a common lisp web server. It uses CL-WHO [Weit03] as the Lisp/HTML markup language, HT-AJAX [Mars07] as an AJAX framework. The combination of these tools with GTFL allows nicely formatted output that the browser can dynamically layout, expand, and contract.

Bibliography

[Loet00] M. Loetzsch. GTFL - A graphical terminal for Lisp, 2000.

Link: http://martin-loetzsch.de/gtfl

[Loet09] Martin Loetzsch, Joris Bleys, and Pieter Wellens. Understanding the Dynamics of Complex Lisp Programs, 2009.

Link: http://www.martin-loetzsch.de/papers/loetzsch09understanding.pdf

[Mars07] U. Marshak. HT-AJAX - AJAX framework for Hunchentoot, 2007.

Link: http://common-lisp.net/project/ht-ajax/ht-ajax.html

[Weit03] E. Weitz. CL-WHO -Yet another Lisp markup language, 2003.

Link: http://www.weitz.de/cl-who/

[Weit06] E. Weitz. HUNCHENTOOT - The Common Lisp web server formerly known as TBNL, 2006.

Link: http://www.weitz.de/hunchentoot