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

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.

Cyril Alberga	Roy Adler	Richard Anderson
George Andrews	Henry Baker	Stephen Balzac
Yurij Baransky	David R. Barton	Gerald Baumgartner
Gilbert Baumslag	Fred Blair	Vladimir Bondarenko
Mark Botch	Alexandre Bouyer	Peter A. Broadbery
Martin Brock	Manuel Bronstein	Florian Bundschuh
William Burge	Quentin Carpent	Bob Caviness
Bruce Char	Cheekai Chin	David V. Chudnovsky
Gregory V. Chudnovsky	Josh Cohen	Christophe Conil
Don Coppersmith	George Corliss	Robert Corless
Gary Cornell	Meino Cramer	Claire Di Crescenzo
Timothy Daly Sr.	Timothy Daly Jr.	James H. Davenport
Jean Della Dora	Gabriel Dos Reis	Michael Dewar
Claire DiCrescendo	Sam Dooley	Lionel Ducos
Martin Dunstan	Brian Dupee	Dominique Duval
Robert Edwards	Heow Eide-Goodman	Lars Erickson
Richard Fateman	Bertfried Fauser	Stuart Feldman
Brian Ford	Albrecht Fortenbacher	George Frances
Constantine Frangos	Timothy Freeman	Korrinn Fu
Marc Gaetano	Rudiger Gebauer	Kathy Gerber
Patricia Gianni	Holger Gollan	Teresa Gomez-Diaz
Laureano Gonzalez-Vega	Stephen Gortler	Johannes Grabmeier
Matt Grayson	James Griesmer	Vladimir Grinberg
Oswald Gschnitzer	Jocelyn Guidry	Steve Hague
Vilya Harvey	Satoshi Hamaguchi	Martin Hassner
Ralf Hemmecke	Henderson	Antoine Hersen
Pietro Iglio	Richard Jenks	Kai Kaminski
Grant Keady	Tony Kennedy	Paul Kosinski
Klaus Kusche	Bernhard Kutzler	Larry Lambe
Frederic Lehobey	Michel Levaud	Howard Levy
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
Jinzhong Niu	Michael O'Connor	Kostas Oikonomou
Julian A. Padget	Bill Page	Jaap Weel
Susan Pelzel	Michel Petitot	Didier Pinchon
Claude Quitte	Norman Ramsey	Michael Richardson
Renaud Rioboo	Jean Rivlin	Nicolas Robidoux
Simon Robinson	Michael Rothstein	Martin Rubey
Philip Santas	Alfred Scheerhorn	William Schelter
Gerhard Schneider	Martin Schoenert	Marshall Schor
Fritz Schwarz	Nick Simicich	William Sit
Elena Smirnova	Jonathan Steinbach	Christine Sundaresan
Robert Sutor	Moss E. Sweedler	Eugene Surowitz
James Thatcher	Baldir Thomas	Mike Thomas
Dylan Thurston	Barry Trager	Themos T. Tsikas
Gregory Vanuxem	Bernhard Wall	Stephen Watt
Juergen Weiss	M. Weller	Mark Wegman
James Wen	Thorsten Werther	Michael Wester
John M. Wiley	Berhard Will	Clifton J. Williamson
Stephen Wilson	Shmuel Winograd	Robert Wisbauer
Sandra Wityak	Waldemar Wiwianka	Knut Wolf
Clifford Yapp	David Yun	Richard Zippel
Evelyn Zoernack	Bruno Zuercher	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))

Chapter 1

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.

Chapter 2

Experiments

2.1 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>
```

2.2 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>
```

2.3 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>
```

Chapter 3

Other work

3.1 Understanding the Dynamics of Complex Lisp Programs [9]

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 uses Hunchentoot [10] as a common lisp web server. It uses CL-WHO [11] as the Lisp/HTML markup language, HT-AJAX [12] as an AJAX framework. The combination of these tools with GTFL [13] allows nicely formatted output that the browser can dynamically layout, expand, and contract.

Bibliography

[1] Jenks, R.J. and Sutor, R.S. "Axiom – The Scientific Computation System" Springer-Verlag New York (1992) ISBN 0-387-97855-0

[2] Knuth, Donald E., "Literate Programming" Center for the Study of Language and Information ISBN 0-937073-81-4 Stanford CA (1992)

[3] Daly, Timothy, "The Axiom Wiki Website"
http://axiom.axiom-developer.org

[4] Watt, Stephen, "Aldor",
http://www.aldor.org

[5] Lamport, Leslie, "Latex – A Document Preparation System", Addison-Wesley, New York ISBN 0-201-52983-1

[6] Ramsey, Norman "Noweb – A Simple, Extensible Tool for Literate Programming"
http://www.eecs.harvard.edu/ ˜nr/noweb

[7] Winograd, Terry "Bringing Design to Software", ACM Press Books, Addison-Wesley Publishing, 1996

[8] Daly, Timothy, "The Axiom Literate Documentation"
http://axiom.axiom-developer.org/axiom-website/documentation.html

[9] Loetzsch, Martin; Bleys, Joris; Wellens, Pieter "Understanding the Dynamics of Complex Lisp Programs" `www.martin-loetzsch.de/papers/loetzsch09understanding.pdf`

[10] Weitz, E. "HUNCHENTOOT - The Common Lisp web server formerly known as TBNL" `www.weitz.de/hunchentoot/`

[11] Weitz, E. "CL-WHO -Yet another Lisp markup language" `www.weitz.de/cl-who/`

[12] Marshak, U. "HT-AJAX - AJAX framework for Hunchentoot" `common-lisp.net/project/ht-ajax/ht-ajax.html`

[13] Loetzsch, M. “GTFL - A graphical terminal for Lisp” `martin-loetzsch.de/gtfl/`