



Development status of CHOMP and Conley-Morse-Database

Using CHOMP and conley-morse-database

Shaun Harker
Konstantin Mischaikow
chomp.rutgers.edu

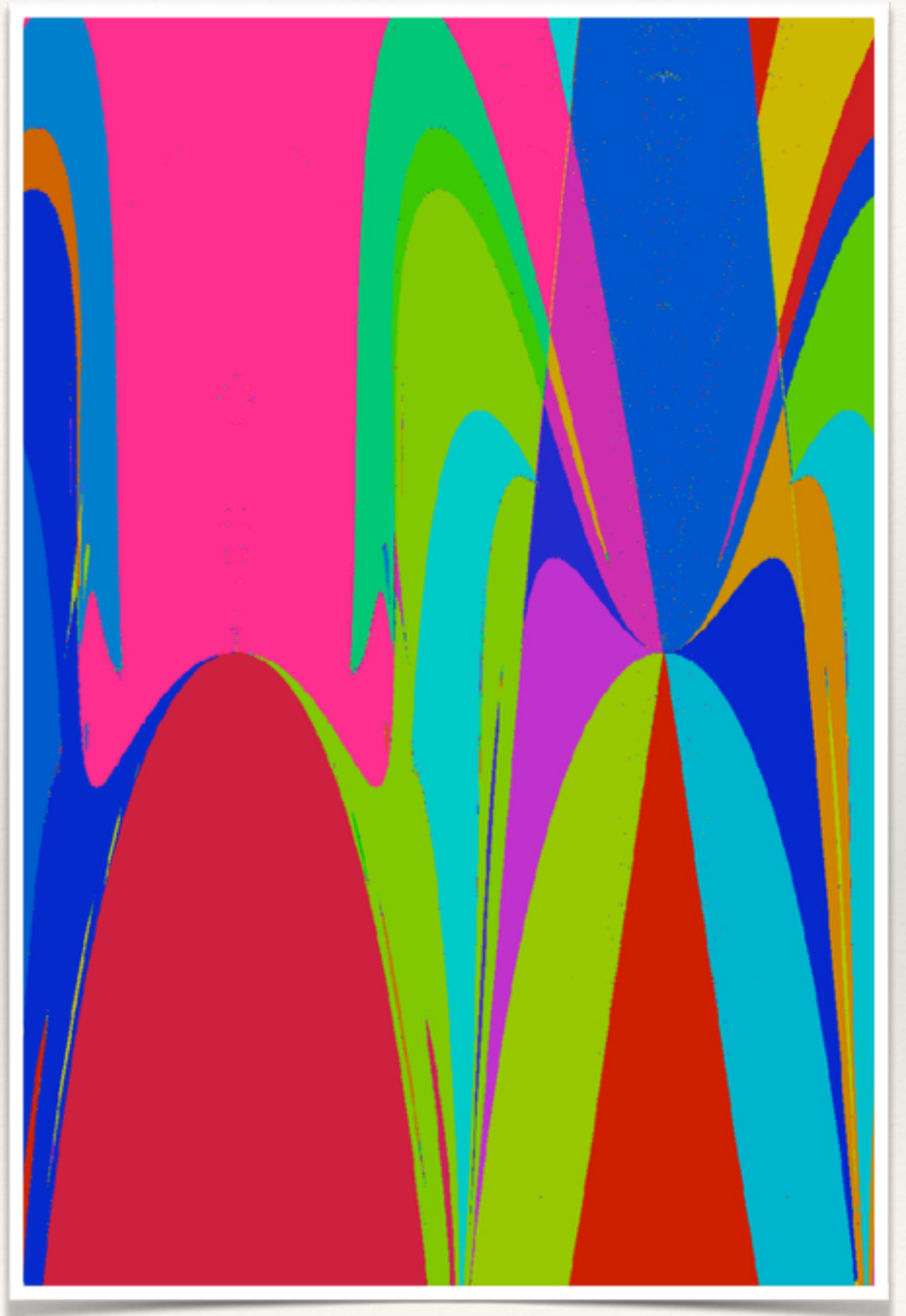
Lorentz Center, Leiden University, August 2014

http://chomp.rutgers.edu/Projects/Databases_for_the_Global_Dynamics/software/LorentzCenterAugust2014.pdf

CHomP and Conley-Morse-Database

Overview

- CHomP project provides tools for computational homology and computational dynamics.
- We'll discuss the status of the software packages CHomP and Conley-Morse-Database and give brief demonstrations



Installation

Installing Conley-Morse-Database also installs CHomP

Conley-Morse-Database is a C++11 program and will require a C++11 compiler such as GCC ≥ 4.7 or Clang. Note that Clang is standard on Mac OS X. To install on Mac OS X (or Linux with root privileges) we could type into the shell:

```
mkdir ~/Work && cd ~/Work
git clone https://github.com/sharker81/conley-morse-database
cd conley-morse-database
./install.sh
```

The prerequisites will install into `"/usr/local."` For Mac OS X, this will install the prerequisites using [Homebrew](#) whenever possible. If brew is not installed, the installer will try to install it; this will require a password.

To install on Linux without root privileges, you need to pick an installation directory you have write access to and install there. For example:

```
mkdir ~/Work && cd ~/Work
git clone https://github.com/sharker81/conley-morse-database
cd conley-morse-database
./install.sh ~/Work
```

CHomP

Two versions:

Original CHomP: features better command line support

CHomP: features discrete Morse theory, used by Conley-Morse-Database

We talk about the second version.

Original CHomP:



The screenshot shows a web browser window titled "Computational Homology Project" with the URL chomp.rutgers.edu/Projects/Computational_Homology/OriginalCHomP/software/. The page features a navigation menu with links for Home, Software, Projects, Education, Links, and People. Below this, a secondary menu includes Software, Introduction, Download, Compilation, Data Formats, Programs, and C++ Library. The main content area displays the CHomP logo in a blue, pixelated font, followed by the text "The Original CHomP Software" and a logo consisting of four colored cubes (yellow, blue, red, green) arranged in a cross pattern. A note below the logo states: "Note: This page describes the original version of the CHomP software, which shares part of the code with the [RedHom](#) project developed by the [CAPD](#) group. A totally [new version](#) of the CHomP software is currently under development." The page then provides a detailed description of the CHomP software package, which includes a C++ programming library and command-line driven programs. It lists several key programs: `chomp`, `homcubes`, and `pprograms`. The page also includes links to various sections: [Introduction](#) (brief introduction to the basic CHomP software), [Download](#) (download page for basic and advanced versions), [Compilation](#) (compilation guidelines for various operating systems), [Data formats](#) (description of text data formats), [Programs](#) (list of programs written in C++), [Chooser](#) (CHomP Software Chooser, an interactive GUI Python script), and [Examples](#) (list of examples contained in the `examples/` subdirectory).

Home Software Projects Education Links People

Software Introduction Download Compilation Data Formats Programs C++ Library

CHomP

The Original CHomP Software



Note: This page describes the original version of the CHomP software, which shares part of the code with the [RedHom](#) project developed by the [CAPD](#) group. A totally [new version](#) of the CHomP software is currently under development.

The CHomP software package consists of a C++ programming library, whose most fundamental features are also made available through a collection of command-line driven programs (pre-compiled version available for some systems). The most basic program that provides access to several homology algorithms (implemented by W. Kalies, M. Mrozek, and P. Pilarczyk) is called `chomp`. In addition to `chomp`, another main program is `homcubes`, which provides a flexible interface for the computation of homology of cubical sets, as well as the computation of the homomorphisms induced in homology by combinatorial cubical multivalued maps. There are also algorithms and programs for computing homology of simplicial polyhedra and chain complexes, as well as for manipulating cubical sets. The following pages describe the contents of the software package in more detail:

[Introduction](#) - brief introduction to the basic CHomP software, which explains some of the most basic concepts and provides instructions on how to use the program `chomp`.

[Download](#) - a download page, both for the basic version, as well as the advanced one; contains the source code and precompiled programs

[Compilation](#) - compilation guidelines for various operating systems; some specific or more precise compilation instructions are also available:

- [Get Started!](#) - very concise yet detailed step-by-step compilation instructions for getting started with CHomP (a PDF file of a handout prepared for the [ICMS 2010](#) in Kobe)
- [For Ubuntu](#) - step-by-step instructions for compiling the CHomP package in Ubuntu (also applies to most Debian-derived Linuxes)

[Data formats](#) - description of the text data formats in which cubical sets, maps, and other related data structures can be saved to and retrieved from files

[Programs](#) - a list of programs written in C++ that are compiled together with the CHomP library

[Chooser](#) - the CHomP Software Chooser, an interactive GUI Python script for composing the command line of most CHomP programs and with detailed information about the `pprograms`

[Examples](#) - a list of examples contained in the `examples/` subdirectory of the source code distribution; these examples illustrate how to use the programs and how to prepare the data; they also indicate some detailed aspects of the software

CHomP (at Rutgers)

sharker81/CHomP

CHomP: Software: Homology

This repository Search or type a command Explore Gist Blog Help sharker81

sharker81 / CHomP Watch 1 Star 0 Fork 0

CHomP -- Computation Homology Project software — Edit

20 commits 1 branch 2 releases 2 contributors

branch: master CHomP / +

passed compiler path flag to cmake

sharker81 authored 2 days ago latest commit fcb5cb1427

examples	initial commit	11 months ago
include	Stopped bundling Cimg.h	6 days ago
source	New cmake build system	6 days ago
tests	Worked on tests	6 days ago
.gitignore	more gitignore	11 months ago
AUTHORS	initial commit	11 months ago
CMakeLists.txt	Fix to find non-standard libraries in install now	2 days ago
COPYING	COPYING preferred to LICENSE	11 months ago
INSTALL	Stopped bundling Cimg.h	6 days ago
README	brought README up to date	6 days ago
install.sh	passed compiler path flag to cmake	2 days ago

Code

Issues 0

Pull Requests 0

Wiki

Pulse

Graphs

Settings

HTTPS clone URL

https://github.com

You can clone with HTTPS, SSH, or Subversion.

Clone in Desktop

Download ZIP

CHomP Overview

- ❖ Computational Homology
- ❖ We'll see a demonstration of command line programs
- ❖ We'll discuss the low-level interface (which gives access to all the features)

CHomP Cubical Complex

Cubical Complex File format: Top dimensional cell locations

```
74 ✓ /Users/sharker/Work/CHomP/examples  
sharker@athena [17:03:41] > more square.cub  
(0, 0)  
(0, 1)  
(0, 2)  
(1, 0)  
(1, 1)  
(1, 2)  
(2, 0)  
(2, 1)  
(2, 2)
```

Cubical Homology Command Line Program

```
sharker@athena [17:03:44] > chomp-cubical ./square.cub  
Betti Numbers: 1 0 0
```

CHomp Simplicial Complex

Simplicial Complex file format: maximal simplices

```
sharker@athena [17:05:34] > more simplex.simp
0 1 2
0 1 3
0 2 3
1 2 3
4
```

Command-Line program for computing homology of
a simplicial complex.

```
sharker@athena [17:05:37] > chomp-simplicial simplex.simp
Betti Numbers: 2 0 1
```

CHomP Chain Complex

We can specify a chain complex via its boundary homomorphisms, which in turn may be specified by the incidence numbers.

```
sharker@athena [17:06:38] > more circle.mat
0
0 0 1
1 0 -1
0 1 1
1 1 -1
```

```
sharker@athena [17:10:11] > chomp-matrix ./circle.mat
Betti Numbers: 1 1
```

low dimension

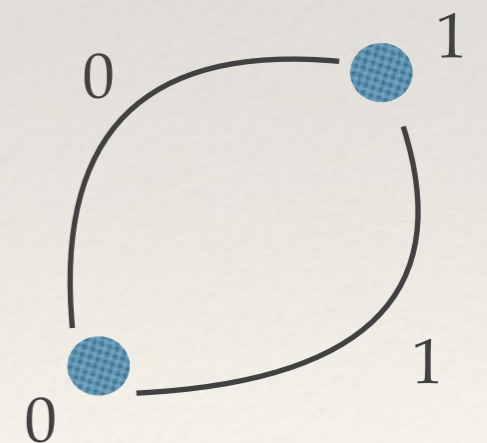
low dim index, high dim cell index, incidence number

low dim index, high dim cell index, incidence number

low dim index, high dim cell index, incidence number

...

repeat for each boundary matrix



$d: C_1 \rightarrow C_0$

CHomp Relative Homology

For cubical complexes, the command line program allows relative homology.

Here is an example:

```
sharker@athena [17:20:02] > more squarewithhole.cub  
(0, 0)  
(0, 1)  
(0, 2)  
(1, 0)  
(1, 2)  
(2, 0)  
(2, 1)  
(2, 2)
```

```
sharker@athena [17:20:06] > chomp-cubical square.cub squarewithhole.cub  
Betti Numbers: 0 0 1
```

CHomP as a Software Library

- ❖ The command line programs are essentially toys; we have emphasized lower-level code interfaces. CHomP has far more features. (Nevertheless we should probably provide more “toys.”)
- ❖ As a software library, CHomP can do more: Induced maps on homology, Conley Index, Graph complexes, discrete Morse theory, novel Complex types, ...

CHomP Complexes

- ❖ Complexes derive from a base class that describes the interface:
- ❖ boundary, coboundary, size, dimension
- ❖ d-Cells are indexed $0, 1, 2, \dots, \text{side}(d)-1$
- ❖ Cells type depends on type of complex.
- ❖ We provide: CubicalComplex, SimplicialComplex, Morse Complex, Subcomplex, BitmapSubcomplex, RelativeComplex, GraphComplex

CHomP Algebra

- ❖ Algebra in CHomP is primarily matrix algebra.
- ❖ We provide a Sparse Matrix class that can give efficient row and column operations (even if they are interspersed)
- ❖ The “Ring” class can be selected. (Potential problem: this is done at compile-time.) We support finite fields \mathbb{Z}_p and also polynomial rings over fields.
- ❖ We provide Smith Normal Form and Frobenius Normal Form
- ❖ Discrete Morse theory also provides some algebraic shortcuts: for example we can solve $dx = c$ where d is a boundary matrix using ideas from discrete Morse theory.

CHomP Discrete Morse Theory

- ❖ CHomP computes homology using either Smith Normal Form or discrete Morse theory (or both).
- ❖ Discrete Morse theory appears to very fast for many practical problems.
- ❖ We compute Morse complex and we may compute chain equivalences that let us move between the Morse complex and the original complex. In particular this allows us to “lift” homology generators in the Morse complex up to the original complex.

Persistent Homology: Perseus


Here is another project using discrete Morse theory for homology.

Perseus: the Persistent Homology Software

www.sas.upenn.edu/~vnanda/perseus/index.html

VIDIT NANDA




Home | Papers | Perseus | Teaching



The **Perseus** Software Project for Rapid Computation of Persistent Homology

Download or Compile

Click on the icon for your operating system in order to download the suitable executable file. The current version is **4.0 Beta**.

Microsoft Windows XP and above	64 Bit	
Linux (Ubuntu, Mint, Debian, SuSE,...)	64 Bit	
Mac OS Tiger and up	64 Bit	

Once you have the file corresponding to your operating system, see "Basic Information and Usage" below. If you'd prefer to compile the software from source yourself instead, keep reading.

The source code is also available as a zipped file [here](#). Download this to a directory where you have read/write permissions. You can now Use any C++ compiler to compile the main file **Pers.cpp**. The choice of compiler depends mainly on your operating system. Microsoft Windows users have various compiler options such as the open-source [mingw](#), or the complete integrated development environment provided by the somewhat pricey [Microsoft Visual Studio](#). Mac users will probably require a hefty [Xcode](#) download and installation on their systems.

If using the [gcc](#) compiler from the command line, just go to the directory with the extracted source files and type:

```
g++ Pers.cpp -O3 -o perseus
```

Of course, you can replace "perseus" in the command above with any executable name of your choice.

Conley-Morse-Database

Conley-Morse-Database Overview

- ❖ Computing Morse Decompositions for a dynamical system
Grids, Strong Components
- ❖ Computing “Clutching” graphs for adjacent parameters
Tree-based Grid algorithms
- ❖ Learning continuation theorems from the clutching data
Union-Find structures
- ❖ Annotating continuation classes with Conley Index information
CHomP
- ❖ Storing the results in a database structure
Handrolled database — switch to SQL?

Configuring Models

- ❖ Conley-Morse-Database requires a “model” directory.
- ❖ This model directory must contain two files:

```
modeldir/  
  config.xml  
  Model.h
```

There is an example “Model.h” file that works unless you are doing something really fancy. In this case you provide “ModelMap.h” which is called by the example “Model.h” file.

config.xml

```
Terminal — bash — 82x31
sharker@athena [18:02:05] > more config.xml
<config>
<model>
  <name>Leslie Model, Depth 12</name>
  <desc> This is a description. </desc>
</model>
<param>
  <dim> 2 </dim>
  <bounds>
    <lower> 19.0 21.0 </lower>
    <upper> 20.0 22.0 </upper>
  </bounds>
  <subdiv>
    <depth> 6 </depth>
    <sizes> 16 16 </sizes>
  </subdiv>
</param>
<phase>
  <dim> 2 </dim>
  <bounds>
    <lower> -1.0 -1.0 </lower>
    <upper> 74.0 52.0 </upper>
  </bounds>
  <subdiv>
    <init> 0 </init>
    <min> 24 </min>
    <max> 30 </max>
    <limit> 10000 </limit>
  </subdiv>
</phase>
</config>
```

The XML file gives settings for parameter space and phase space. The settings for phase space control some of the algorithms.

We can provide a name and description of the model in these first fields.

Dimension and Bounding Boxes

```
Terminal — bash — 82x31
sharker@athena [18:02:05] > more config.xml
<config>
<model>
  <name>Leslie Model, Depth 12</name>
  <desc> This is a description. </desc>
</model>
<param>
  <dim> 2 </dim>
  <bounds>
    <lower> 19.0 21.0 </lower>
    <upper> 20.0 22.0 </upper>
  </bounds>
  <subdiv>
    <depth> 6 </depth>
    <sizes> 16 16 </sizes>
  </subdiv>
</param>
<phase>
  <dim> 2 </dim>
  <bounds>
    <lower> -1.0 -1.0 </lower>
    <upper> 74.0 52.0 </upper>
  </bounds>
  <subdiv>
    <init> 0 </init>
    <min> 24 </min>
    <max> 30 </max>
    <limit> 10000 </limit>
  </subdiv>
</phase>
</config>
```

We specify upper and lower bounds for phase space and parameter space. The number of entries should match the “dim” setting.

Param settings

```
Terminal — bash — 82x31
sharker@athena [18:02:05] > more config.xml
<config>
<model>
  <name>Leslie Model, Depth 12</name>
  <desc> This is a description. </desc>
</model>
<param>
  <dim> 2 </dim>
  <bounds>
    <lower> 19.0 21.0 </lower>
    <upper> 20.0 22.0 </upper>
  </bounds>
  <subdiv>
    <depth> 6 </depth>
    <sizes> 16 16 </sizes>
  </subdiv>
</param>
<phase>
  <dim> 2 </dim>
  <bounds>
    <lower> -1.0 -1.0 </lower>
    <upper> 74.0 52.0 </upper>
  </bounds>
  <subdiv>
    <init> 0 </init>
    <min> 24 </min>
    <max> 30 </max>
    <limit> 10000 </limit>
  </subdiv>
</phase>
</config>
```

“Depth” means number of subdivisions — so 2^{depth} boxes across.

Sizes means “number of boxes across” repeated for each dimension

When conflicting, “sizes” settings overrides “depth” setting (which is the case here)

(init,min,max,limit)-scheme

```
Terminal — bash — 82x31
sharker@athena [18:02:05] > more config.xml
<config>
<model>
  <name>Leslie Model, Depth 12</name>
  <desc> This is a description. </desc>
</model>
<param>
  <dim> 2 </dim>
  <bounds>
    <lower> 19.0 21.0 </lower>
    <upper> 20.0 22.0 </upper>
  </bounds>
  <subdiv>
    <depth> 6 </depth>
    <sizes> 16 16 </sizes>
  </subdiv>
</param>
<phase>
  <dim> 2 </dim>
  <bounds>
    <lower> -1.0 -1.0 </lower>
    <upper> 74.0 52.0 </upper>
  </bounds>
  <subdiv>
    <init> 0 </init>
    <min> 24 </min>
    <max> 30 </max>
    <limit> 10000 </limit>
  </subdiv>
</phase>
</config>
```

Computation of Morse
Decompositions is
controlled by (init, min,
max, limit)-scheme
which is set here.



Hierarchical Morse Decomposition Algorithm

- ❖ Given a grid we construct a directed graph for the dynamics and compute the recurrent sets.
- ❖ We then subdivide those recurrent sets (given certain criterion are met (see next slide)) and repeat.

(init,min,max,limit)-scheme

- ❖ init: unconditional number of subdivisions the phase space grid undergoes to begin
- ❖ min: the minimum resolution of a recurrent set.
- ❖ max: the maximum number of subdivisions a grid element may be subdivided
- ❖ limit: the size threshold at which we decide not to further subdivide a recurrent set that has been subdivided \geq min times, but $<$ max times

ModelMap.h file

- ❖ You need to write the code for the dynamical system.

```
class ModelMap : public Map {
public:
    typedef simple_interval<double> interval;

    // User interface: method to be provided by user
    // Parameter variables
    interval p0, p1;

    // Constructor: sets parameter variables
    void assign ( RectGeo const& rectangle ) {
        // Read parameter intervals from input rectangle
        p0 = getRectangleComponent ( rectangle, 0 );
        p1 = getRectangleComponent ( rectangle, 1 );
    }

    // Map
    RectGeo operator () ( const RectGeo & rectangle ) const {
        // Convert input to intervals
        interval x0 = getRectangleComponent ( rectangle, 0 );
        interval x1 = getRectangleComponent ( rectangle, 1 );

        // Evaluate map
        interval y0 = ( p0 * x0 + p1 * x1 ) * exp ( -0.1 * ( x0 + x1 ) );
        interval y1 = 0.7 * x0;

        // Return result
        return makeRectangle ( y0, y1 );
    }
};
```

parameters

constructor

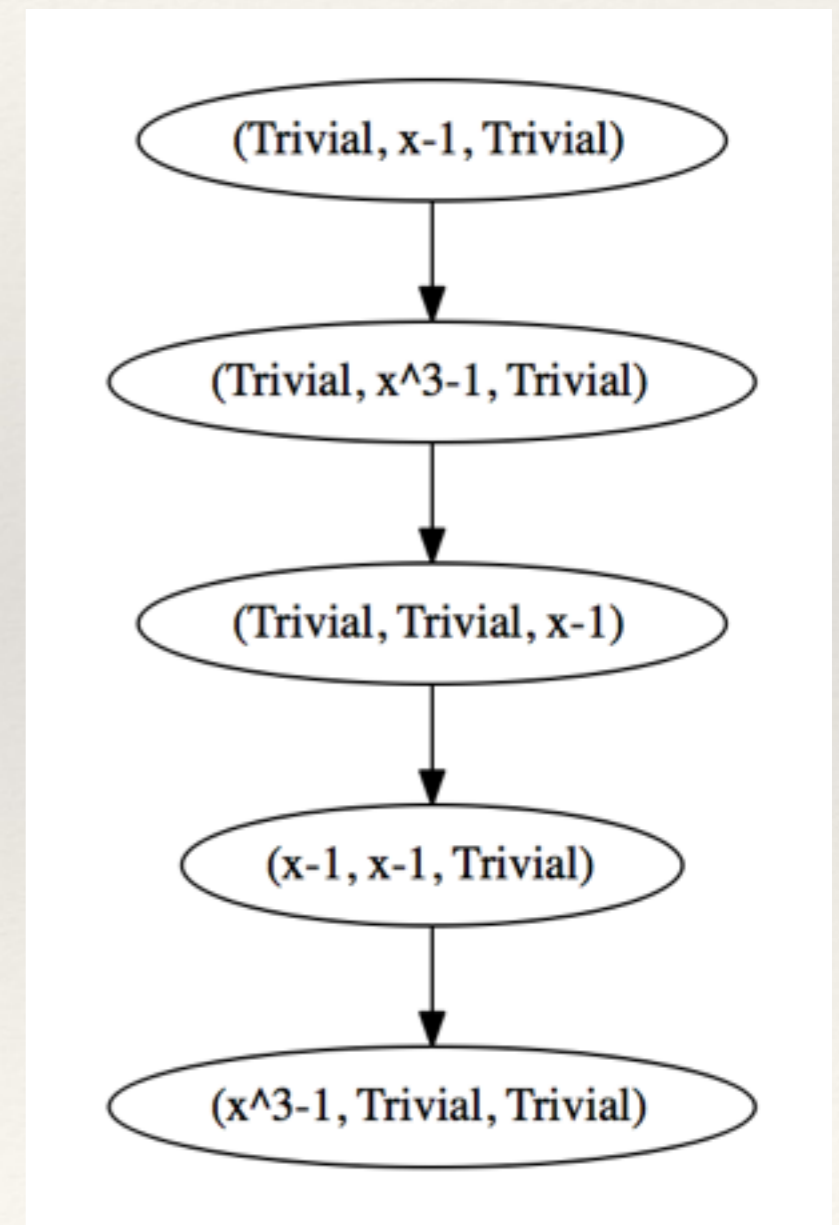
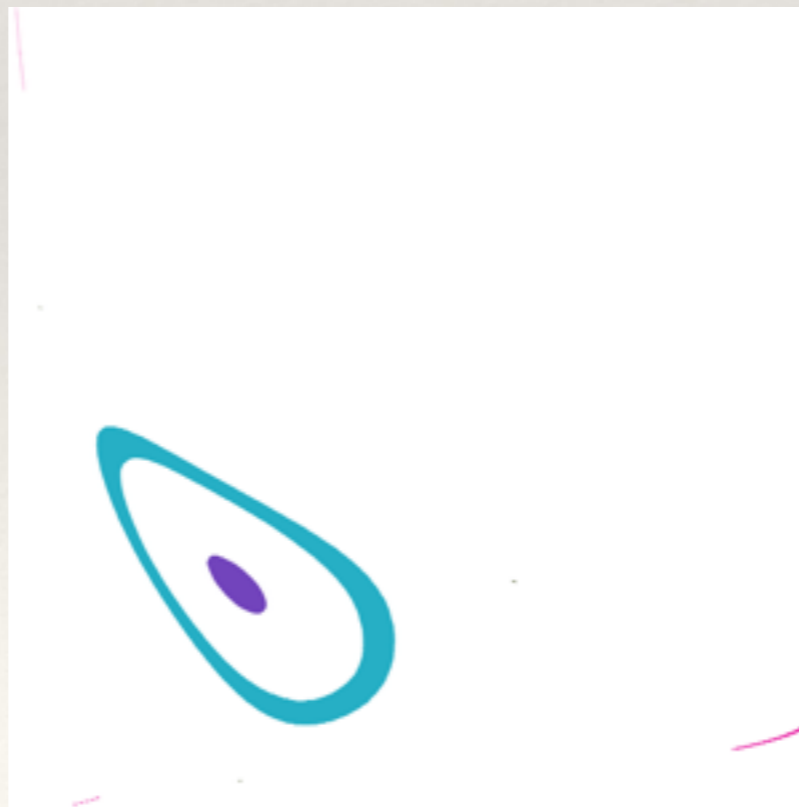
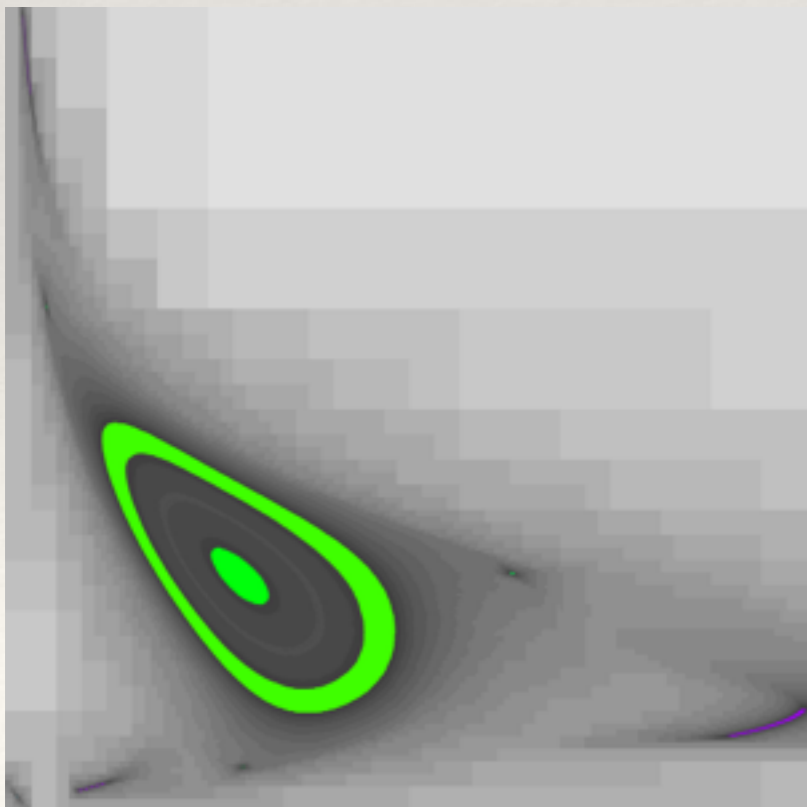
dynamical map

CMDB SingleCMG

We compute the Leslie2D example at an exact parameter value $(20.0, 20.0)$ with $(\text{init}, \text{min}, \text{max}, \text{limit}) = (0, 24, 30, 10000)$

The SingleCMG program can be compiled and run as follows:

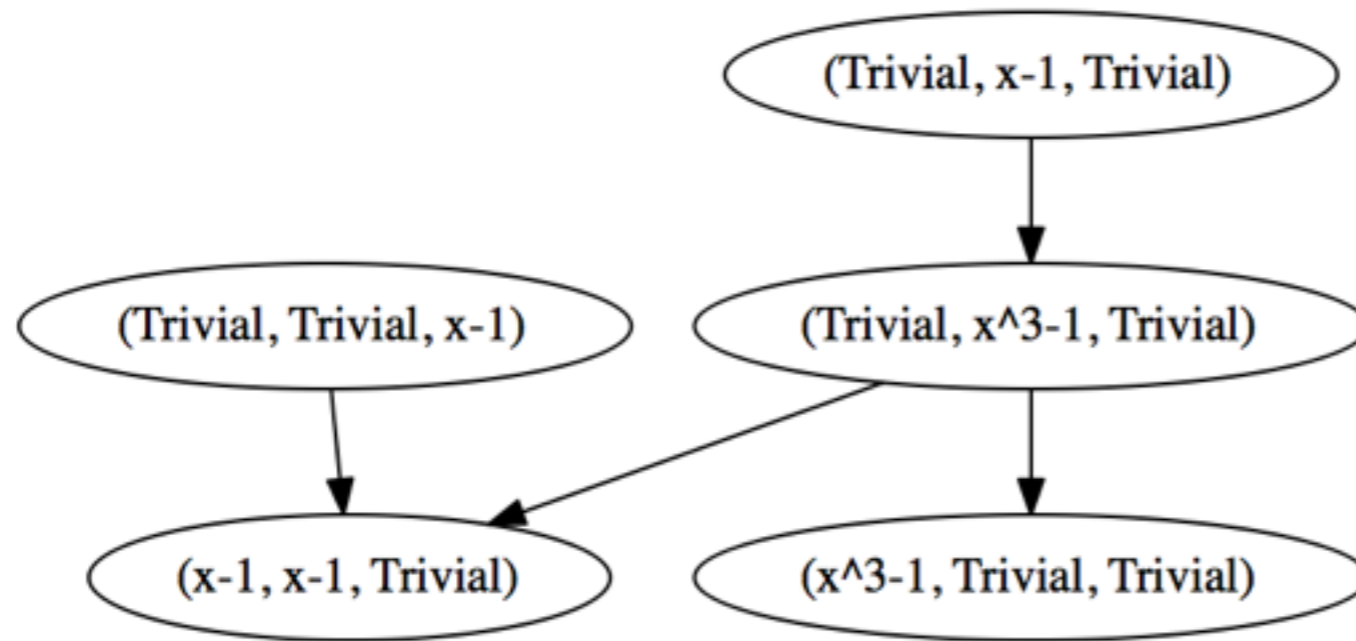
```
cd conley-morse-database
make SingleCMG MODELDIR=./examples/Leslie2D
cd ./examples/Leslie2D
./SingleCMG ./ 20 20
```



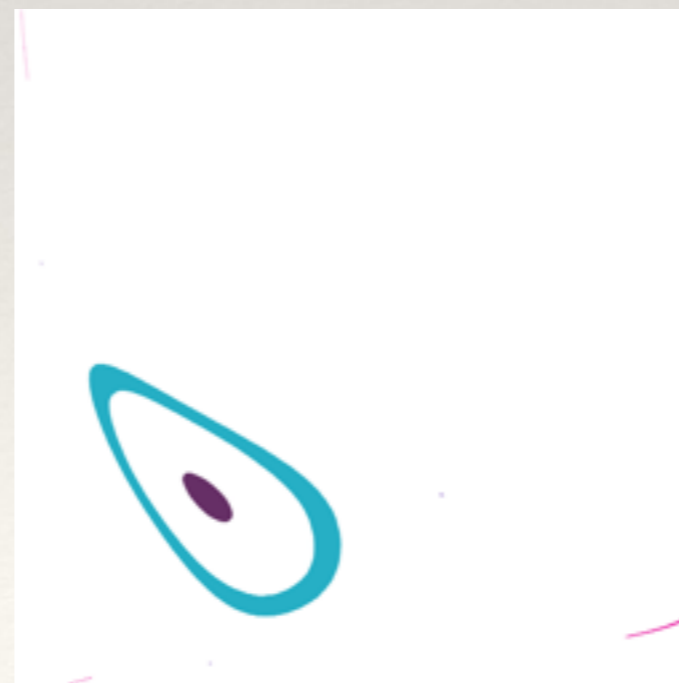
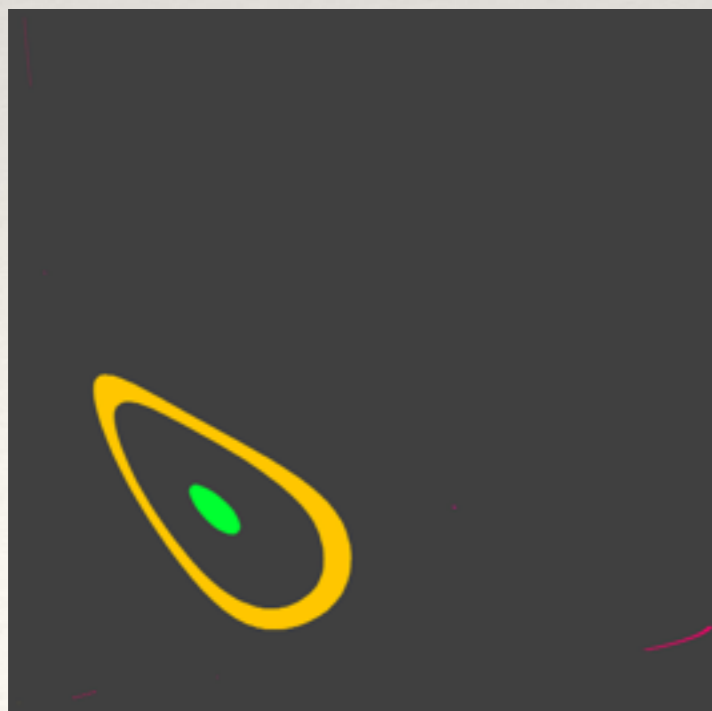
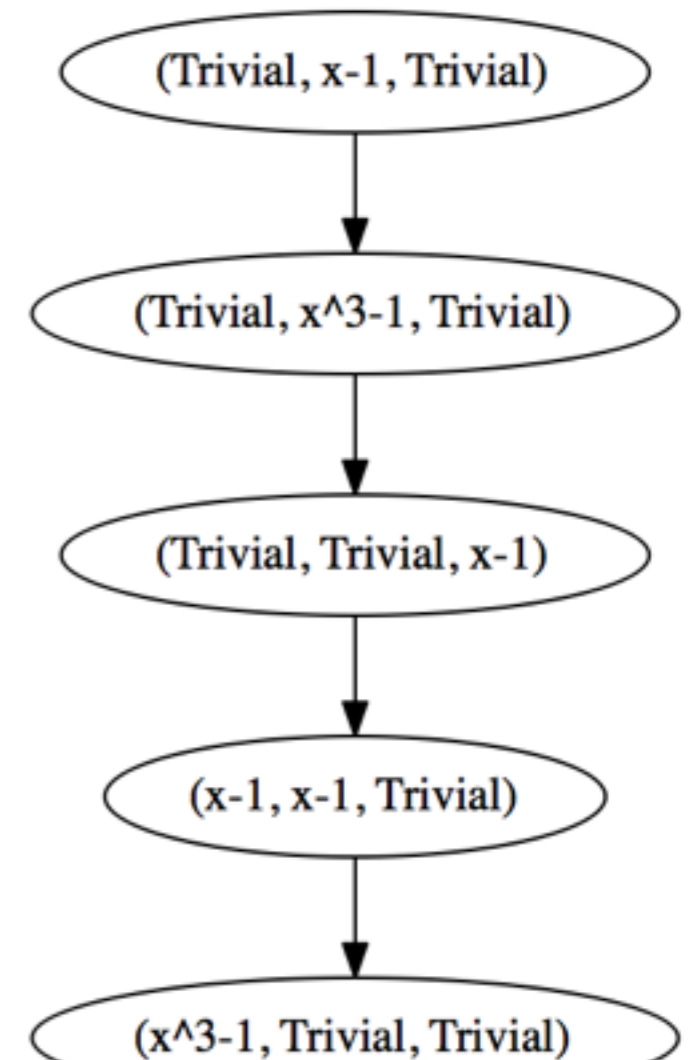
Monotonicity Problem for Reachability

- ❖ The partial order of the last example was not too great. It contained “false positives.”
- ❖ This has to do with limitations of the hierarchical scheme. We cannot recompute at the end on all grid elements because of a “monotonicity problem” — what if, as an artifact arising from the numerics, the images of smaller rectangles are not strictly contained in the images of larger rectangles? We haven’t yet found a completely satisfactory solution to this.
- ❖ A brute force solution is to push the “init” setting way up (but this is far more expensive!)

We compute the Leslie2D example at an exact parameter value $(20.0, 20.0)$ with $(\text{init}, \text{min}, \text{max}, \text{limit}) = (24, 24, 30, 10000)$



compare to previous result using $(0, 24, 30, 10000)$:



CMDDB – the main program

To run on a cluster:

```
cd conley-morse-database  
./CMDDB ./examples/Leslie2D
```

CMDDB calls “*script.sh*”. Edit this to choose number of cores or change command line arguments.


To run on a single machine:

```
cd conley-morse-database  
make MODELDIR=./examples/Leslie2D  
cd ./examples/Leslie2D  
mpirun -np 8 ./Conley_Morse_Database ./
```

Number of logical processor cores



Command line parameters are passed to Model.h. Currently the first one is reserved to be the path at which config.xml is to be found.



CMDDB – Program Design

- ❖ The program operates in three phases which we call the Morse Process, the Continuation Process, and the Conley Process. The program can be compiled to only perform some of these steps. This can be handy if something gets interrupted and needs to be restarted, but you don't want to completely start over. It also can come in handy if the Morse process isn't finishing, since we can actually continue on using the checkpoint files that are created.

```
#####  
# makefile for Conley-Morse Database project  
#####  
COMPUTE_MORSE_SETS := yes  
COMPUTE_CONTINUATION := yes  
COMPUTE_CONLEY_INDEX := no
```

Morse Process

- ❖ The Morse Process computes the Morse decompositions and the clutching relations between them for adjacent parameters. The output of this phase is a file called “database.raw”



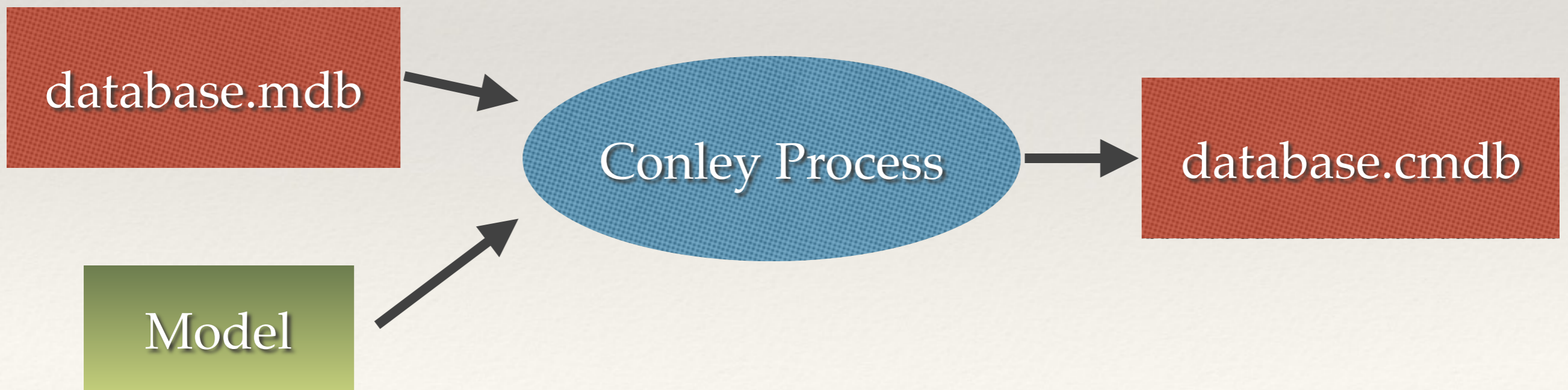
Continuation Process

- ❖ The Continuation Process takes “database.raw” as input and produces a “database.mdb” file containing continuation classes but getting rid of the clutching relations. (The philosophy is that we care about the clutching relations only insofar as we can produce continuation theorems from them.)

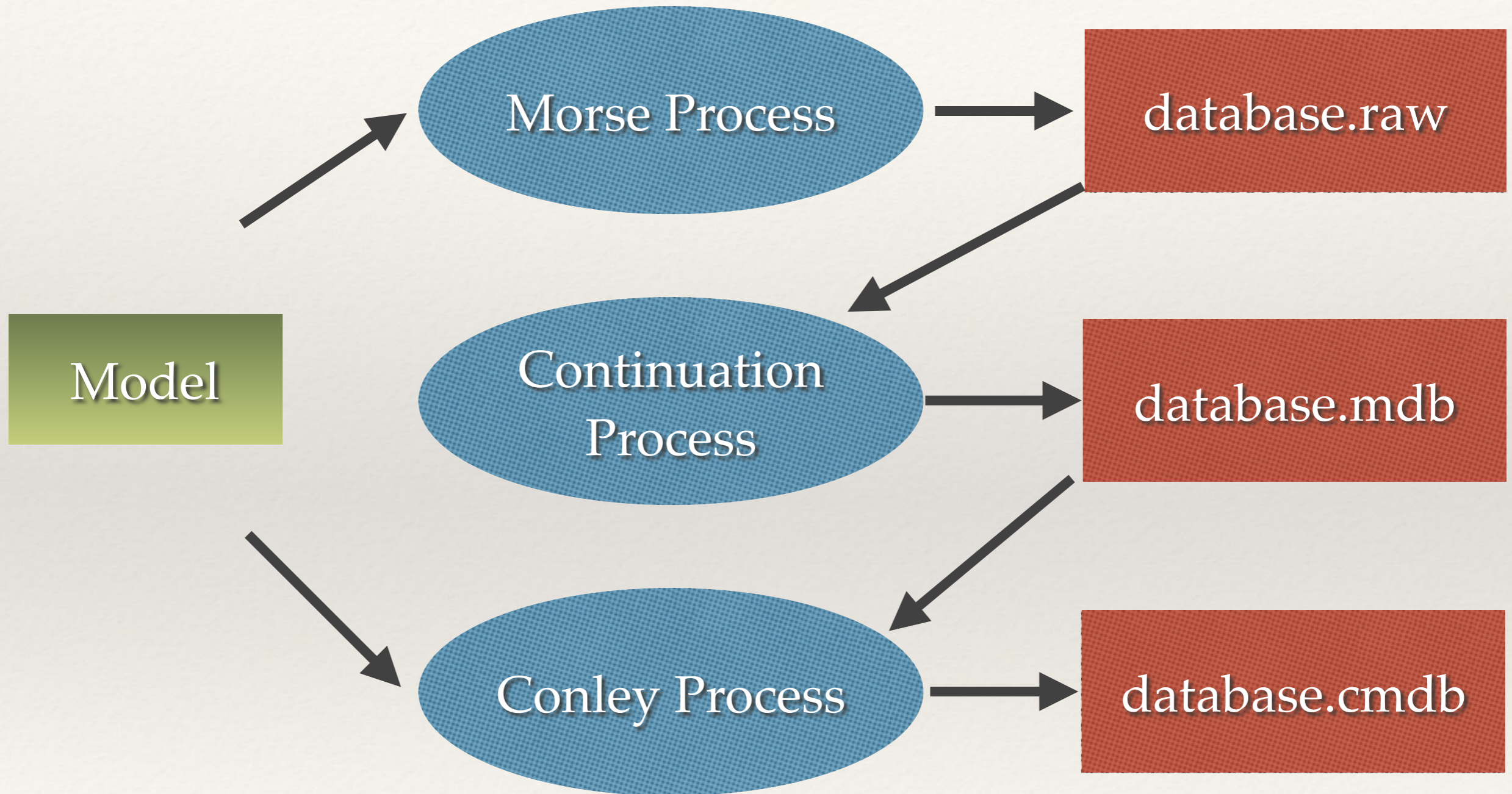


Conley Process

- ❖ The Conley Process loads “database.mdb”, chooses a representative from each “isolating neighborhood continuation class,” computes Conley Index information, and then annotates the set with it. The output is stored as “database.cmdb”

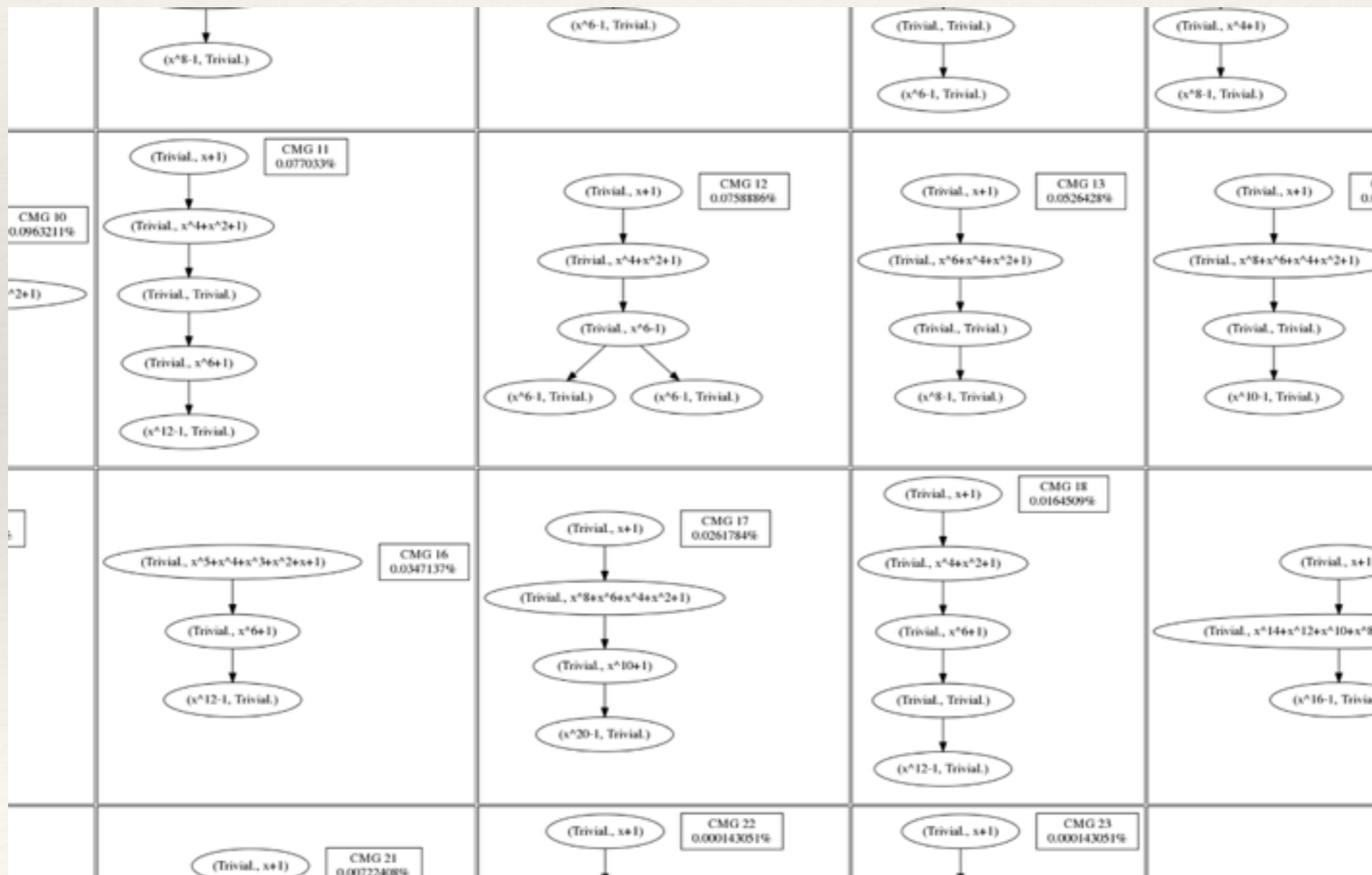


Overall:



CMDDB Zookeeper

Extra program: produces a “Zoo” of Conley-Morse-Graphs, Hasse diagrams, Conley Indices in HTML format.



CMDDB Database-Explorer

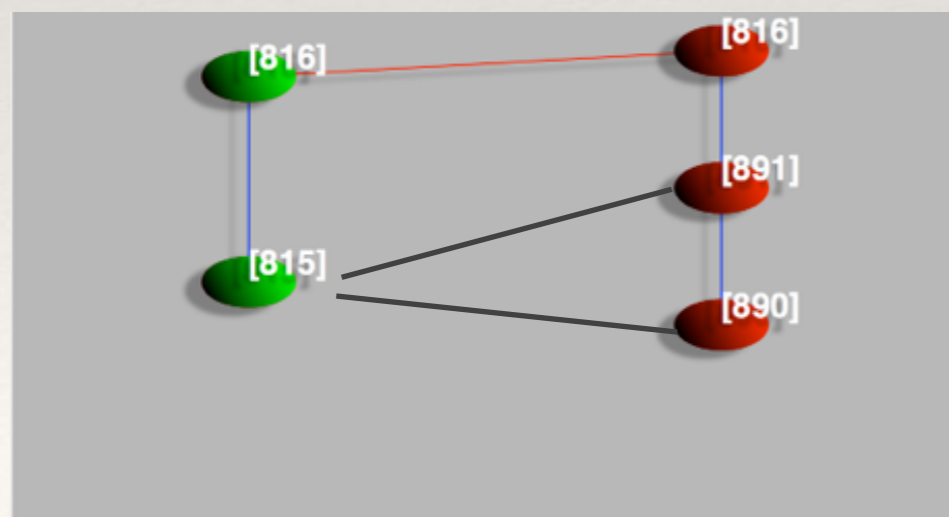
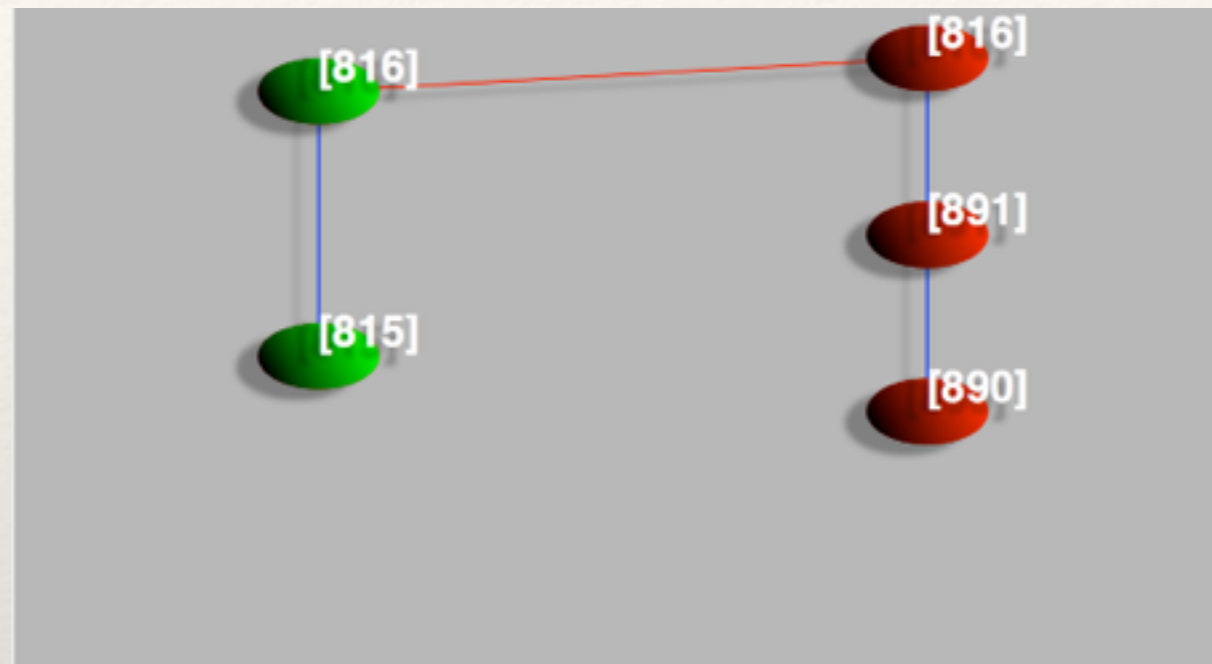
Database-Explorer is a Mac OS X Application that loads “database.mdb” or “database.cmdb” files through a dialogue window at startup.

The screenshot displays the 'database-explorer' application window, which is divided into four distinct views:

- Continuation Graph Window:** Shows a network of nodes connected by lines, representing the continuation graph. The nodes are colored in various colors (blue, green, yellow, red, purple, orange).
- Parameter Space Window:** Displays a complex, multi-colored region representing the parameter space. The axes are labeled as (Z)(X)[-1.00, 1.00] x (Y)[0.00, 3.14].
- Dialog Window:** Contains text describing the Conley Index of INCC for three different cases:
 - Conley Index of INCC 890:
Dimension 0: $x^4 + 2$
Dimension 1: Trivial.
 - Conley Index of INCC 891:
Dimension 0: Trivial.
Dimension 1: $2x^2 + 2$
 - Conley Index of INCC 816:
Dimension 0: Trivial.
Dimension 1: $2x^1 + 2$
- Morse Graph Window:** Shows a vertical sequence of three red nodes labeled [816], [891], and [890], connected by a vertical line.

You can click on various entities which results in different views being shown.

For example, clicking on an edge in the continuation graph results in the display of two Morse graphs, and the continuation theorems that are known between the combinatorial Morse sets. Notice that the numbering of the combinatorial Morse sets reflects which continuation class it is in.



The clutching probably looked like this, but this information is not available except in “database.raw”.

CMDDB Data Structures and Algorithms

- ❖ The processes are distributed on HPC clusters using the cluster-delegator software package.
- ❖ We provide Succinct Grids which have very low space use.
- ❖ We also have a space efficient version of Tarjan's algorithm.
- ❖ Parameter space is dealt with very abstractly; in fact we can generalize to parameter graphs where each vertex tells us how to instantiate a dynamical system and an adjacency tells us that both vertices provide outer approximations for some system.
- ❖ We provide a Grid class that can handle Atlases. However, we have not yet interfaced this new Grid class to CHomP.

Development Directions

Suggestion: Git Repositories

Git is nice since:

It lets you use a forking workflow rather painlessly:

1. Fork another user's project
2. Make a branch
3. Make changes, add files, etc...
4. Merge in changes from the main branch that occur (rebase)
5. Perform a pull request and the project owner performs a code review and decides if he wants your changes.

Future directions for CHomP

- ❖ Support for Conley Index and induced homology of maps independent of conley-morse-database
- ❖ No repetition of code between CHomP and conley-morse-database for map evaluation
- ❖ Low-level interface for obtaining homology without requiring generators.
- ❖ AtlasComplex for (stratified) manifolds

Future Directions for CMDDB

- ❖ Clutching when the phase space bounds vary
- ❖ Isolating neighborhood classes for intervals in Morse graph, not just singletons.
- ❖ Index Pairs for multi-scale grids.
- ❖ Resolve “monotonicity problem”
- ❖ Better support for ODE time-t maps.

Future Directions for CMDDB, cont

- ❖ SQL-style databases for querying and scaling
- ❖ External memory computation of equivalence classes
- ❖ Portable database-explorer (web based?)
- ❖ Query Language
- ❖ A richer “Zoo” web interface

Thanks