Nitrous Alife Bomb

Scott Draves

February 21, 1996

1 intro

Nitrous, Alife, Bomb: a web spanned by interactive graphics, semantics-based program transformation, artificial life, and philosophy.

this web is an attempt to explain the graphics that i do, how they work, why i do it, what they mean, their relationship to my computer science research (ie my job), and how it all connects to a mess of philosphical meanderings and excerpts/reviews about artificial life and complexity. much of it is no more than my personal notes on popular science books i read. my contribution to the growth of exhibitionism.

Contents

1.1 this document

This document was preduced with the markup system. It reads its own format and produces HTML and Latex versions.

The HTML version of this document is available at `http://hopeless.mess.cs.cmu.edu:8001/nab/nab.html`. Postscript from Latex at `http://www.cs.cmu.edu/ spot/nab.ps`.

1.2 Feedback

If you have corrections, suggestions, criticisms, extensions, pointers, additions, substractions, rejections, opinions, references, or any comment at all, please feel free to visit the HyperNews response form

2 graphics

i like to make computer generated visual effects. this includes still images, animation, realtime screen hacks, eye candy, visual muzak, digital video feedback, mathematical visualization, algorithmic collage, etc. In general i work by writing programs and creating their datafiles; the final pixels are always produced automatically. Still jpeg images, mpeg animation, and source code are all available on the web at my online gallery `http://www.cs.cmu.edu/ spot/images.html`. If you haven't seen the pictures yet, don't bother reading this.

These sections describe the programs i've written and how they work:

- flame iterated function systems (IFS)
  - math behind IFS
- color cmaps, how i build them
- fuse scrambled images
- slip pixel flow
- bomb combination of above plus interactivity and CAs

3 related work

John Whitney *Digital Harmony: On the Complementarity of Music and Visual Art* ISBN 0-07-070015-X

dave tristram's electro-paint.

After Dark and various screen saver archives.

Karl Sims, also his SIGGRAPH91 paper *Artificial Evolution for Computer Graphics*.

Evolutionary Art and Computers; Todd, Steven and William Latham; Academic Press, Harcourt Brace Jovanovich, Publishers; 1992; ISBN 0-12-437185-X

3.1 flame

flame is the name i use for my iterated function system (IFS) software. it's primary purpose is to create *shape*.

The basic idea of IFS is that from a handful of simple functions on the plane (eg a 3x2 matrix, or the complex log function), one can generate a picture. Every set of functions defines a different picture. By continuously varying the parameters, the picture can be animated. The shapes appearing in the pictures are fractals, strange attractors in particular.

My interest in iterated function systems (IFSs) goes back to freshman year when Bill Poirier showed me what he called 'recursive pictures'. For a graphics class I explored generalizations of this idea, including higher dimensions.

Several implementations later I gave up on a tree of recursive functions producing geometry, and adopted the randomized IFS algorithm (see below), which gives a stream of points in the plane. Just drawing these points produces a crude fractal image. this is how the interactive flame explorer, the xlock mode, and (more or less) the bomb flame mode work.

I produced my first quality monochrome images by collecting a high-resolution (typically 3000x4500) 2D-histogram of these points, taking its point-wise log, and then filtering down to 1000x1500 (the aspect ratio of 35mm film). Because fractals have such detail and texture, correct filtering is essential. The addition of color (see below) resulted in some nice pictures. These images required about four hours each to generate on a 100Mhz R4000.

That much is easy, the hard part of making an IFS picture is picking the functions! Because I was programming with C, this problem naturally broke into two stages: parameterizing a portion of function-space, then picking the coordinates of the desired functions. [connect to nitrous, explain parameterization] Like most people, I started out with affine functions (3x2 matrices). To produce pictures with greater variety of character, the matrix may be composed with a 'variation'. Over many months I developed six variations (libifs.c:109):

linear none

sinusoidal component-wise sine

spherical project around unit circle $(1/\bar{z})$

swirl rotate by radius

horseshoe rotate by angle $(z^2/|z|)$

fireworks polar transform $(c_0 + c_1 \ln z)$

bent piecewise linear (C^1 discontinuities on axes)

Rather than parameterizing with an integer specifying one of these, a vector in R^7 specifies an linear combination. This has the advantage of being continuous. Thus each function has 6+7=13 parameters.

The above parameterization contains only the tiniest sliver of function space, but already the user is confronted with as many as 75 dials to set in order to specify a picture. Furthermore, the relationship between the parameters (dials) and the picture is entirely non-invertible (as is the nature of complexity). So how can one produce interesting pictures?

I chose brute force: generate many random parameter sets, render them very quickly at low-quality, and save any good ones that appear. It was pretty easy to collect 200 that I liked. To guarantee that the matrices are contractive, their entries are chosen in the biunit cube.

this departs from the traditional creative process by factoring it into spew and filter phases. filtering is easy because it's multiple-choice: you only have to *recognize* what you want. what makes a good spew? diversity and density. see here.

The next stage was animation: because the parameter space and the mapping from parameters to pictures are continuous, a picture can be put in motion by varying its parameters. Furthermore, any two pictures can serve as end points for interpolation, creating an animation that smothly transforms one into the other. I've made one high-quality animation (written frame by frame to video-disc, facilities provided by NTT-data), and several low-quality MPEGs.

The method of interpolation is important: element-wise interpolation of matrices produces more degenerate matrices (zero determinate) than necessary. Such matrices usually don't produce good pictures since they destroy information. An alternative is to break the 3x2 matrix into a 2x2 matrix, and a translation. Interpret the 2x2 matrix as a pair of complex numbers, and interplate them in log-space, and interplate the translation linearly. This feels like a very natural way to interpolate matrices, does it have a name?

An easy way to produce an animation from a single picture is to rotate it: just compose rotations onto each of its functions. This tends to preserve some nature of a picture while utterly transforming it. Furthermore, the transformation is circular: after rotating 360 degrees, the picture returns to its original state. Combining rotation with other matrix operators (scale, translate, spread eigenvalues, etc) might be a good way to provide interactive control of the high-dimensional parameter space.

For the realtime versions of flame (in xlock and bomb), I wanted the image to change continuously and indefinitely, without ever repeating itself. To accomplish this, the matrix parameters follow follow a Lissajous figure in the biunit cube. Each matrix entry is assigned its own random frequency, then the frequencies are collectively normalized to bound the speed of the path in parameter space. The variation-combination parameters remain fixed.

Currently I am working on a After Effects plug-in for flame animation.

3.1.1 The IFS Algorithm

Mathematically, an IFS is defined by a finite set of functions $\{f_i\}, 0 <= i < n$ on a space S ie $f_i : S \rightarrow S$. These functions (typically between two and five) define a subset $P \subset S$ by the following recursive set equation: $\bigcup_i f_i(P) = P$ P is the recursive picture. But given the functions, how can one solve the equation and produce the picture?

Note that if $x \in P$ then $f_i(x) \in P$, thus if we have *some* of P, then we can get *more* of P. And not just one more point: so too are $f_j(f_i(x)), f_k(f_j(f_i(x)))$ etc in P. To get *all* of P we must consider every possible string $ijk...$ of applications. Thus the IFS algorithm

```
draw point x
pick random i
x = f[i](x)
repeat
```

works because a string of random digits will, in the limit, include every possible sequence of digits as a subsequence.

But how do we get the first point in P? As long as the functions are *contractive*, then the above algorithm will work given *any* x because P is an *attractor*. You only have to throw out the initial dozen or so iterations while x settles into the fractal.

furthermore, this gives us more than just membership in the set, it gives *density*: how likely are we to see the point (or one very near it) as the algorithm runs?

3.1.2 Coloring an IFS

a monochrome histogram viewed with a colormap works ok (bomb uses this), but it's possible to do better.

add another dimension to the IFS, and map this value. This coordinate is independent of the others, and its functions are very simple: the first function of the set averages the color value with zero, the other functions average it with one. The colormap value of a point is then simply the base-two interpretation of the string of function choices made by the IFS algorithm, a number between zero and one (the most recent digit has most significance). libifs.c:98

The advantage of this scheme is that the coloring is *structural*. The disadvantage is that the histogram must now be three channels (RGB) instead of one. Note that the log only applies to intensity, so the RGB values are transformed into YUV coordinates, and the log is applied to Y only.

an alpha channel has also been implemented.

3.2 colors

this section describes cmaps, and explains how i derive them automatically from scanned images.

before i go any further, i must rant: i'm sick of seeing otherwise very nice computer generated pictures, and fractals in particular, ruined by something so easy to fix: a rainbow colormap. at least people have stopped using FULLY SATURATED RAINBOWS!!! and you cynics don't believe in progress...

a colormap is a palette plus an ordering. usually a cmap is just a table mapping 8 bit indices to 24-bit colors. since the cmap is often used to visualize a continuous scalar parameter (eg heat), ideally the mapping from number to color is continuous. that is, colors for adjacent numbers are near each other.

i generate cmaps automatically from scanned images. i've been using the same cmaps for several years. most are derived from nature backgrounds (and a few foregrounds) to pictures found on alt.sex.binaries.erotica. some are derived from scanned impressionist paintings. the derivation has two stages: first it picks the colors, then it sorts the colors to make the cmap continuous.

the palette of 256 colors comes from picking 256 random small rectangles (say 4x4 pixels) in the source image, and taking the average of each of them. a randomized hill-climber improves the ordering. see here.

3.3 fuse

The fuse algorithm was inspired by gnu-emacs dissociation mode and the movie Society. There are several variant forms, but they share this basic idea: start with a set of scanned source images, break them into many small tiles (typically 25x25 pixels), and then build an output image by assembling the tiles in a way that is locally visually coherent.

All the variants are built from the same fundamental operation: associative image matching. This match function takes a key image tile and searches the source images for a matching tile (minimizing sum of squares of pixel difference). how matching works.

So far I have three variants. Most of the high quality images on my web page are made by growing the output image from the center out. The output starts all black, then one seed tile is blitted to the center. The output image grows by picking tiles on the edge, half in the output image (the overlap) and half in blackness. The source tile that matches the overlap best is blitted into place, thus extending the image.

The second variant tries to reconstruct a template image from the source tiles. As such, the match function tries to minimize both the difference between the tile and its context, and between the tile and the template. A blending function combines these two optimization criteria. The face made out of parts images were made like this.

The third variant is used by the bomb program. It doesn't grow from a seed, but uses whatever is in the framebuffer and continuously modifies it. It picks a random tile on the screen, finds the best match for it, and blits the match to the screen. The result has a bit of positive feedback to it, and has a tendency to go all white or all black. Basically, it still needs a lot of work.

3.3.1 match

The fuse images are all built by associative image matching. This match function takes a key image tile and searches the source images for a matching tile (minimizing sum of squares of pixel difference).

The match for a key tile is found by picking a collection of random candidate tiles (say 10000) in the source images, comparing each of them to the key, and returning the most similar tile. This allows easy speed/quality trade-off. Similarity is measured by subtracting the two tiles, then summing the squares of these differences (In RGB space. Perhaps better would be to weight intensity) (this is called diff-dot (what's the real name?)).

In order to improve the memory access pattern, the tiles are compared in scan-line order. Typically the number of candidates is larger than the height of the source images (say n times larger). So we still get a good distribution just by looping over the scan lines, and picking n tiles on each line.

To run much faster, the interactive version uses a multi-level algorithm: all the source images are prefiltered down by 5, and the search takes place with 5x5 tiles instead of 25x25 ones. The top few matches are saved, and compared again at full resolution to find the 'best'. Of 10000 candidates currently it saves 10000/25 = 400 tiles for comparison at full rez.

3.4 slip

This is a very simple idea (developed in 1989 on a sun4/110 using pixrect, Sun's first and last good graphics software), but it's quite effective because it's cheap to compute. It operates directly on the frame buffer via many blit operations. Each blit moves a rectangle of pixels (typically 20x20) by a small distance, usually just one pixel. The effect of these blits is to cause the pixels to flow, as if they were colored oil droplets suspended in swirling water.

But which of the eight one-pixel directions? There are several algorithms I've used for chosing the direction at each pixel. Each of these defines a flow-field:

- random. brownian motion.
- one of several fields defined by simple equations, eg spiral and eggbeaters.
- start with a grayscale image. Compute its gradient, then rotate the gradient 90 degrees. As the pixels swirl, eventually the original image becomes visible. Other transformations of the gradient would also probably be interesting...

There are many more possibilities. So far I have not tried time-dependent fields.

There is a slight difficulty in implementation: the flow-field generally contain non-integer values, but the blits are only one pixel. If the low-bits are just rounded-off the resulting flow-field is too coarse. The problem is essentially the same as dithering an image, and I adopted a solution from that domain: stochastic rounding. That is, instead of rounding 0.1 to 0 deterministically, there is a 90% chance (per pixel) 0.1 will be approximated with 0, and 10% chance with 1.

The above is a poisson update algorithm. I also have a synchronous implementation for high-quality animations. It uses a bresenham algorithm at each pixel, thus besides the delta vector provided by the flow-field per pixel per frame, each pixel maintains an error vector as state. Each frame the delta is added to the error (the deltas are normalizes so the largest delta is less than one). If pixel A's error vector strays into adjacent pixel B, then B's color is copied into A, and the error is reduced in that direction.

Finally, some additional blits are sometimes added to change the topology of the frame buffer from a square into torus.

3.5 bomb

bomb is *automatic interactive visual stimulation*.

bomb is a program that runs on a PC. it belongs to the vaguely defined class of graphics programs called 'eye candy', or 'screen savers', or 'screen hacks'. it produces continually changing visual effects independently, or under keyboard control. bomb is work in progress: my ultimate goal is to create a visual musical instrument.

the implementation for OpenGL, linux, and PowerMac is available here, including a collection of screen shots.

the program has many modes, each of which may produce several (or many) effects. most of the effects are non-representational textures, harmonic geometry, and fractals. a library of scanned images and characters provides a visual iconic symbolic component to some of the modes. the icons function as lyrics.

the texture effects are produced with cellular automata (CA): the prime CA used is cooled diffusion with a mask and underflow. 'diffusion' means that each pixel is (say) heat, and is averaged with its neighbors each frame. this creates the blurring and spreading. 'underflow' means that when heat reaches -1, it goes to its maximum. this creates the chaotic boiling. 'with a mask' means that a bitwise constant is chosen, and all heat values are ANDed with this constant. (persian rug from hopkins, mask from jepler).

this heat layer is then driven by various functions:

- underflow-enable is driven with a square wave
- another CA such as life, fredkin's rule, symmetric life, or an excitation-relaxation system (barking dogs/neurons/forest fire/BZ).
- another heat layer at a different time/space scale
- an image
- a trail of points from random-walk sinusoids produces the rotor (idea by tcl)
- a trail from a time-varying flame IFS
- a collection of fish or slugs (by nix)
- more complex things that distort the diffusion kernel as a fn of the driver, producing flow and fiber effects.
- symmetries (tiles and mirrors) are used both for visual effect and to make computing the driving function faster.

a separate mode implements reaction-diffusion textures. it has two reactants, four slightly different reactions, and a collection of different diffusion patterns:

- isotropic
- horizontal
- vertical
- square grid
- zig-zag
- image-picks horizontal/vertical
- radial
- circular
- gradient(image) rot by pi/2 (still experimental)

basically i've thrown together a menagerie of parameterized screen hacks (some of them contributed by others) with a zillion keyboard commands for tweeking the parameters and an automatic navigator that can set and adjust the parameters on its own (aleatoric). the fundamental kbd command is on the space bar: it picks a fresh random parameter set, killing the current set.

4 research

This is a summary of my thesis research in progress. Compared to the proposal i submitted to the department lightweight languages for interactive graphics, this presentation is shorter, less technical, more conversational, more colloquial, and a bit flakier. also see Compiler Generation for Interactive Graphics to appear in PE96.

objective the problem: generality vs performance

examples example: RTCG of power procedures

metaphor languages and RTCG

past work existing approaches to RTCG

partial evaluation the language approach

nitrous my system

results what graphics can we do with it?

chips how chip architecture fits in

telephone the telephone operator principle

4.1 problem and objective

The objective of my research is to develop software systems for building complex interactive graphics environments. Consider these 'power-user' applications:

quickdraw windows, framebuffers, video io, scan conversion, character generation.

photoshop defines new brushes and filters on-the-fly.

renderman 3D geometry, motion, and the shading language.

doom-tv over a LAN, with live video streams mapped onto walls and characters.

Simsian CAs mutating cellular automata and video streams for user selection.

Current software practice is inadequate for such systems. Here's the essence of why: in graphics there are many pixels. on one hand, you want the per-pixel routines (ie the bits of code that actally draw each pixel on the screen) to be flexible to support user-specified operations. On the other hand, making the per-pixel code flexible and general kills performance by putting conditionals in a loop. this is the fundamental performance vs generality trade-off at work.

When prototyping, exploratory programming, evolutionary programming, toolkit building, and 'builder' building, this problem is exacerbated because this is when software is at its most flexible.

The standard technique in this situation is batching (aka buffering and loop inversion), which uses temporary memory to avoid the inner conditionals and increase the bandwidth. unfortunately batching is hard to program and increases latency. but latency is critical to interactivity. (see proposal)

The only solution is run-time code generation (RTCG). here, each time the user requests a new operation, a loop is generated just for that operation. thus the conditionals are eliminated, no temporary memory is required, and latency is uneffected.

the problem-axis of my thesis then becomes: 1) how to support RTCG in a programming system? and 2) how can RTCG be used in interactive graphics?

4.2 example

before i present my and other approaches to RTCG, here's an example:

in graphics the `power` function is a good candidate for RTCG because when shading a surface `power` is called with a different base per pixel, but the exponent remains constant as it depends only the the surface properties.

```
double power(double base, int exp) {
   int t, bit = 1;
   int square = base;
   double result = 1.0;
   while (bit <= exp) {
      if (bit & exp)
         result *= square;
      bit = bit << 1;
      square *= square;
   }
   return result;
}
```

the simplest implementation:

```
exponent = surface();
for (i = 0; i < bign; i++)
  ans[i] = power(exponent, base[i]);
```

can then be replaced with:

```
exponent = surface();
p_e = power_gen(exponent);
for (i = 0; i < bign; i++)
  ans[i] = p_e(base[i]);
```

where `power_gen` is a generating extionsion—it does the RTCG. the constant (static) computation has been factored out of the loop, leaving only the dynamic computation. the control flow is done once, leaving only a block of multiplications in the loop.

4.3 metaphor

here's where languages enter the picture: a compiler for a language translates programs into executable form, ie performs code generation. so one way to support programming RTCG would be to support *compiler generation* (and make sure they work at 'runtime').

by adopting language labels, the last code fragment becomes:

```
program = user();
objcode = compile(program);
for (i = 0; i < bign; i++)
  ans[i] = objcode(argument[i]);
```

which precisely describes program developement: the programmer edits the program, compiles it, then compiled code runs many times (note it may only be *some part of* the compiled code that runs many times, and these are probably the only parts worth compiling).

`power` is an *interpreter* for a language of integer exponents. just like a typical lisp eval, the program directs control flow which effects computation of the argument. if the program is fixed, then this control flow can be performed once by the compiler, rather than repeated every time the program is run.

it's a stretch to say `power` is an interpreter with integer programs. but if we actually look at shading in 3D graphics, we find the per-pixel programs vary from a handful of numbers to lists of structures (light and surface properties) to shade trees to Renderman programs. the transition to 'real' languages is smooth.

but what use is the metaphor? if we want to create functions that create code at runtime, why relabel the problem 'compiler generation'? answer: because compiler generation is a classic problem in computer science, and recent results in semantics-based program transformation appear promising for our purposes. it suggests using interpreters to *specify* generating extensions. furthermore, the formalism of languages provides a unified approach to the usually vexingly unspecified and irregular meta-programming issues.

so if each instance of RTCG is specified with a language/compiler, then a system where RTCG is cheap+easy to program is one where it's cheap+easy to create languages/compilers—one where languages are lightweight.

there are reasons to believe that techniques for 'lightweight languages' may be more broadly useful than just interactive graphics: embedded application languages, human-computer interfaces, network protocols, and software engineering module interfaces share the same fundamental problems, and may too benefit from lightweight languages.

4.4 existing systems

this section summarizes past and existing RTCG practice.

originally RTCG was implemented by writing machine code that produced machine code eg [blit] [synthesis]. this is hard.

the lisp community has long valued RTCG. backquote/comma and `syntax-rules` make writing code-producing code easier. common lisp's standardized macro and meta-programming features are considered crucial to fully utilizing lisp [onlisp] [abelson].

RTCG is a foreign term, not a foreign concept: the lisper thinks 'compiling eval' instead.

the C community's has been slower to address RTCG and meta-programming: the current state-of-the-art is cpp, batch compilation, and lex/yacc, with the occasional sed or awk script for good measure. C libraries for building and compiling Intermediate Representations (IRs, eg graphs, quadruples, RTL, etc) are appearing [pardo] [DCG]. more recent projects augment C with backquote-style syntax for RTCG ['C] [spin/chambers thing]. i would find something like these systems very useful.

analysis: backquote is very general but still confusing. automatic 'transformer generators' (eg synax-rules, lex/yacc) are useful, but lack the generality of backquote. how can a transformation system be both general and automatic?

maybe this will work. note that comma/backquote indicate *types* of computation: early and late (different from value types like `int` or `char*`). thus type-checking can be adopted to avoid 'obvious' logical problems with backquote code, and type-inference can automate the placement of some commas/backquotes.

these 'computation types' (static and dynamic, unquoted and quoted), are known as 'binding times'. they are used in 'partial evalution', the subject of the next section.

4.5 partial evaluation

this section introduces self-applicable partial evaluation, the semantics-based program transformation i use.

The most recent attempts to 'automate backquote' are the self-applicable partial evaluation systems exemplified by Similix. The heart of these systems is `cogen`. cogen transforms interpreter source code into compiler source code. it does two things for the programmer: a binding time analysis (BTA) divides the program into static and dynamic 'binding times', and a generation phase handles the bookkeeping necessary to produce loops, avoid infinite loops and assemble code fragments.

Applying Similix to the power example, the programmer builds a generating extension `comp` by feeding power's source text and a *binding time pattern* to `cogen`:

```
(cogen 'power '(static dynamic) "power.sim")
(comp (list 10 '***))
(load-residual-program)

(power-0 2) → 1024
```

Since Similix wasn't built for RTCG and because Scheme's programming model is so undefined, the interface has several glaring weaknesses: files, generated symbols, top-level state, etc. but these are cosmetic ills (solved by fabius, see below).

the real drawback of PE is that cogen requires more than just an interpreter with binding time pattern. to avoid pathalogical behavior (non-termination, excessive specialization, bad separation, etc) the interpreter must be annotated and tweeked to stay within the confines of cogen's analytical powers. perhaps this *brittleness* outweighs the benefits of automation.

4.6 Fabius

Mark Leone is building a compiler called Fabius that uses BTA to support Lightweight RTCG. It works like cogen, but its compilers produce machine code directly in one pass, rather than producing Scheme (like Similix) or an intermediate language (like ‘C). So compilers run fast (aka are Lightweight), but produce weak code. It solves Similix's cosmetic problems, but portability is reduced.

4.7 my system

My system is called nitrous [note about name]. I have tried to take the best ideas from ‘C, Fabius, and Common Lisp, and build a compromise system.

I use an Intermediate Representation like ‘C, to allow optimization of the generated code, and to make the system more portable.

I use a BTA like Fabius to automate the division of the program into static and dynamic portions.

From Common Lisp i take the reflective idea of defining the programming environment and making it available at runtime. Nitrous supports multiple code transformation systems, and allows them to be composed. In particular, the output of a generated compiler may be fed back into cogen. this allows multiple layers of languages to be compiled (see here).

4.8 preliminary results

readers who like graphics are probably wondering if they've been tricked: what does PE have to do with DOOM? so far i've only addressed one end of my problem axis: how to support RTCG, but not how to use it in interactive graphics. This section makes the connection; the proposal explains how the code for some of these might work.

- quickdraw
- photoshop
- renderman
- Simsian CA
- max dsp

4.8.1 quickdraw

consider quickdraw. video enters and leaves the machine through framebuffers, each with its own format. some images are grayscale, some are bitmaps, and some have indexed color. each pixop or scan-converter is implemented for each format, burdening the programmer and wasting memory.

instead, a language describing operations over generalized images could create scan-converters specialized to particular formats. the same line-drawing or bitblit code could work with RGB, RGBA, sequential, grayscale, indexed, and maybe even single-bit formats.

a simpler example is the 8-way symmetry of Bresenham's algorithm. ordinary lisp macros (or exotic cpp hackery) suffice for implementation since (i'm guessing) there's little advantage to not compiling all the cases. however, partial evaluation may simplify the implementation by allowing the programmer to convert a variable into a constant (in this case, the variable is which of the eight cases applies) just by a single annotation.

the previous example represents an interesting alternative use of cogen: rather than writing highly general procedures, write very small procedures. after a routine is debugged and identified as a candidate for optimization, the annotation is added. since the transformation is semantics-preserving, there is less that can go wrong, and it is easier to debug.

another application for specialization is rectilinear ops, that is the case where one component of a delta vector is zero. a line or rectangle routine can run much faster if its arguments are known to be so aligned, generating code for this case requires somehow making this logical relation static [note vs-on-line] [note binding-time-improvement].

how general can the image representation be? how general a space of image operations can be supported? a union image hands off pixel operations to one of its children images with a translation, depending on the location. this is the building block of the macintosh feature where the desktop spans several monitors, and windows may cross framebuffer boundaries. in theory, once window placement is known, alignment and overlap tests may be eliminated, but it remains to be seen if nitrous is actually the right tool.

4.8.2 photoshop

consider photoshop (or even more so, fractal paint). it comes with a large collection of brushes and effects, many of which are complex enough to have their own control panels and user interfaces. you can write your own plug-in modules, but it's too hard.

instead, consider a brush language where user-controls, brush-motion, and image arithemetic are highly integrated. now when the user feels the need for something new, she copies the closest existing brush, pops it open and edits the program, clicks ok, and tries it out. the experiment is safe because the brush language is safe.

consider the efficient implementation of filters, or more generally any image function where each output pixel is a function of several near-by input pixels—a neighborhood, perhaps 3x3 or 5x5. say your target machine has registers and cache. a sliding window increases register reuse, and tiling the path of the window increases cache use [Wolf]. the problem is this loop is much more complex than the obvious 4-nested-loop implementation.

the fortran approach hides the complexity in an optimizing compiler, the transformation happens automatically. this introduces brittleness and requires the compiler to search.

another path is to use a macro to abstract the window/tile pattern. my objective is to provide easy ways of writing such macros.

4.8.3 renderman

consider a renderman modeler. the user opens a dialog and edits a shader program. the program is compiled for better interactive performance as the geometry is dragged about. the user decides to adjust the some parameters to the shader. when she drags a slider the shader is specialized so that only values dependent on that slider are recomputed—the static sliders are 'program'. [usoft].

on another front, the 'display list' or 'modeling hierarchy' can be converted from an interpreted data structure into executable code as needed.

4.8.4 Simsian CA

consider Simsian evolution of cellular automata on a PC. a genetic algorithm produces local pixel programs. the neighborhood functions combine with buffers to produce video streams. the user selects pleasing programs for reproduction by the genetic algorithm. the task of the programmer is then to create a language which mutates well.

4.8.5 dsp

Max is a data-flow graphical programming system for MIDI streams (named after Max Matthews, the father of computer music. the last time i saw it was 1990, when Andy Schloss was using it with 'digital drumsticks'). Do that but for audio samples and video pixels. This might be a system like Danenberg's Actic music language, but suitable for realtime interactive performance.

4.9 chips

the transition from CISC to RISC involved a transfer of responsibility from hardware to software: addressing modes, scheduling, delay slots, procedure call, etc. were implemented by the compiler rather than with microcode.

say we are deciding if a given feature should be handled by software/compiler or in microcode/hardware. if the information required to handle it isn't known at compile time, it must be left to the hardware. with RTCG one can know more at compile time, thus RTCG is more useful on RISC chips than on CISC.

but RISC isn't the limit: with RTCG sometimes cache/buffer access patterns, data alignment, loop lengths, shift distances, etc are available at compile time. thus Digital Signal Processing (DSP) chips (which replace cache with directly accessed on-chip SRAM and have no alignment circuits) may make an ideal target for RTCG. RISC and DSP both have three levels of memory hierarchy (multi-ported registers, on-chip ram, off-chip ram), but in RISC the cache manages the on-chip ram, whereas DSP leaves it to the programmer/compiler. bypassing the dynamic cache circuits saves some time, but managing it in software is difficult. compare this to the CISC to RISC transition and the appearance of register allocation.

in fact DSP chips are exactly those often used for interactive graphics systems like video games, 3D rendering, and of course audio filtering and effects.

but DSP isn't the limit: Field Programmable Gate Arrays (FPGAs) are a new kind of chip that allow their logic to be reconfigured in the field—at run time! in the same way one writes ‘ROM’ into an EEPROM, one burns custom boolean circuits into an FPGA, thus reprogramming the hardware. these may provide the ultimate target for RTCG [IseSa93].

4.10 telephone

Bill Scherlis taught me the telephone operator principle: when the telephone system was first being built Bell hired lots of women as telephone operators. as the system grew, and as people used it more, Bell hired more and more people. when management projected the number of operators required into the future, they saw that in not so many years, they would have to hire everyone in the country! ridiculous? in fact it happened: now we all dial our own numbers.

we can see a similar thing happening if we look at the number of programmers. of course, not everyone is going to sit down and code in C, but I believe some level of automation will enable everyone to program...

5 alife

here are some references to start out with:

- alife faq
- santa fe
- the fractal faq
- comp.ai.genetic faq

i have bookmark lists on complexity convergence.
but i can’t help but rant:

- what is life?
- what is complexity?
- what is vitality?
- what is randomness?
- Bateson on Life
- Schrödinger on Life
- Gell-Mann on Complexity
- connection to graphics
- layers
- progress

- The Great Chain of Being
- comments on *From Complexity to Perplexity*
- Gould's Natural History
- eschaton and omega
- what is god?

some stuff not categorized yet:

something stops being interesting if you can learn to predict it. something is alive if you can't predict it. `bateson.html`. his definition of life is relative: a super-being may be able to 'play croquet' with something we consider alive—even ourselves.

some people use the word 'emergence' to talk about life. a system is alive if 'new' behavior emerges from a system whose rules don't code that behavior 'directly'. how can we make a science of such 'emergence' if we can't measure it objectively?

we draw a qualitative semantic barrier between two levels of a system if the connection between them is too 'complicated' for us to understand, even though the basic rules on either end are understood. this is the barrier against which classic scientific 'reductionism' hits.

personally i try to parameterize a very large space, and risk producing some duds. when bomb is used interactively, duds would be considered 'bad notes' or errors; hopefully the user recovers and moves on. in auto-mode, when the system is improvising on its own, you must suffer with them. working on auto-mode is an attempt to change coordinates on the parameterization so that 'interesting' areas are easy to find. a first step in this direction is automatic detection of 'low variance' images, and rejecting them. thus we have returned to the search for a objective (ie exists as a program i can run) definition of noise. for example, we might do well to reject images that don't have 'pink' frequence-power distributions.

> Natural selection is a mechanism for generating an exceedingly high degree of improbability. — Sir Ronald Aylmer Fisher (1890-1962)

> There are no differences but differences of degree between different degrees of difference and no difference. William James

[digression could you build a living system (one with OEE) that had a small parameter space? what if you introduced a meta-level: imagine a simple CA supporting a complex meta-behavior such as wire-world or neuron-models. but the large parameter space isn't eliminated, it just moves up to the initial conditions for the CA.

so say the True laws of physics are just simple equations. we know they support life, but only just the right initial conditions are alive: empty space is dead.... or maybe you need a lucky sequence of quantum improbabilities. or maybe you just have to wait a very long time (best estimate: 20 billion years).]

5.1 life

what is artificial life (alife)? more to the point: what is life?

some people claim that 'alife exists'—that we have already created software systems 'complex enough' to warrent the label 'life'. the real achievement here is defining the axis from life to non-life, rather than the systems so far.

because really, the 'liveliest' software systems to date are still only simple shadows compared to even bacteria. but animism is popular: a virus is alive, the sun is alive, the telephone system is alive, a hive has 'a life' beyond the sum of the lives of its members. why not the latest GA? because they're not complex enough? where do you draw the line?

[expansion: compared to nature, the programs that people write are tiny, non-complex, and have little content. when comparing bacteria or cells to computer programs, sometimes people add up the RNA information content and compare this to some source code, and conclude that software has equaled nature. but this leaves out the molecular state of the organism, a huge resource (though it is quite random, thus has a large discount). [link difference between human layers (eg computer languages) and naturally evolved layers (eg cells, human language)]]

in An Evolutionary Approach to Synthetic Biology Tom Ray defines life with a check-list of two: self-replication and 'open ended evolution'. he considers evolution to be the fundamental process of life. but most of us would say a turing-successful AI is alive because it can *think* and *learn*, even though it lacks replication . afterall, darwinian evolution is only one of many learning algorithms.

furthermore, you and i are certainly alive, but our brain's resources and capacity for learning are both finite (though very large). it's not even clear that life on this earth or in this universe is open ended (though it's certainly very very large).

my definition of life is based on *complexity* and *emergence*.

Schrödinger answered *What is Life?* with *Living matter evades the decay to equilibrium*, see here.

Gell-Mann's definition is based on transmission of information, see here

Nell Tenhaaf's Simorg Culture

> The central premise of a-life ... is that codified knowledge operating as rules (heuristics of search algorithms, in computerese) can parallel the necessary self-organization as Darwin theorized it

more.

isbn 0-942299-29-9; Zone 6; 'incorporations' ed Crary, Kwinter Manuel DeLanda 'non-organic life'; 909.82 I37 (hunt 2nd floor)

p138 "Roughly, we could say that phenomena of self-orgaanization occur whenever a bifurcation takes place: when a new attractor appears on the phase portrait of a system, or when the system's attractors mutate in kind."

cf MST

5.2 complexity

how to measure complexity? given a particular behavior or phenomena or message, how do you measure its information content? how objective can this definition be made?

Shannon used probability/thermodynamics to define content as the negative-entropy: $-\Sigma \lg p_i$ of the message. but what predicts the probabilities? the message "101010101010..." has little content but high entropy. If we 'changed coordinate systems', replacing "10" with "2", then our message "222222..." has the right size. One could define the content by the size of the output from `gzip`, or any other data compression algorithm. This is a step in the right direction, but which algorithm is Right? What is the content of the first 100000 digits of π (3.1415...)?

Kolmagorov solved this by counting the size of the algorithm: the complexity of a message is the size of the smallest program that can create that message. so digits of π can be compressed into a small program that uses a mathematical series. but which language do we use?

Meta System Transition Theory (as outlined in the PCP) claims to have objective definitions of complexity and even progress, but so far i haven't found them.

but they all have the same problem: noise appears to be complex. how can meaning be made objective? how can we write a program to find something if we don't know what it will be?

what is the difference between different and random? it is meaning, and this is subjective. how can you show that noise is *not* a message from aliens?

a noise-generating program might be very small, but it won't generate the 'right' output—it may appear similar. what is the meaning of a photograph of sand on the beach? does it include fragments of histories of rocks and waves? what about something 'truly random': radiation? what if i encrypt the picture, but you don't have the key? is that noise?

my bookmarks on the subject are here.

5.3 vitality

say we find a program on the net, and we want to how well it supports life without looking at its source code. so we run the program, and something lifelike appeared on the screen. there are several possibilities:

maybe it's just conway's life started with a particular configuration that produces a particularly long, varied, lifelike run, but any change to the seed ruins the plant.

or maybe the program simply plays a time-lapse video recording of real cells.

or maybe we just got lucky: it just generates random shapes, and we happened to be looking when the army of typing monkeys did something intelligent.

the real question is: how large a variety of interesting behavior can the program produce, and how often does it produce interesting behavior in that space? see here. i call this property *vitality*.

eg in boids a school/flock emerges from a couple of plausible local behavior rules. in conway's life, a squirmy behavior emerges, but only one behavior. after looking at either of these systems for a while, one gets bored. interesting and novel events don't happen often enough.

how wide a range of different behaviors can the system create? tierra's GA creates a host of strategies, but do the same basic strategies appear every time you run the program? i don't know, i haven't tried it. how alive is a system one must run many times (or wait a very, very long time) before the truly interesting and exotic appear?

the point of Stephen Gould's book *Wonderful Life* is that in the evolutionary history of life on Earth, every time you run the program, you would get wildly differing results. that is: real life is very vital (see here).

Gell-Mann talks about depth and crypticity.

the other question (how likely are duds) is a classic cosmological/philisophical problem, see The Anthropic Principle.

5.4 randomness

Paul Vitányi paulv@cwi.nl writes about randomness.

P. S. Laplace (1749 - 1827)

Samuel Johnson

etc.

5.5 Bateson on Life

from *Metalogue: Why Do Things Have Outlines* by Gregory Bateson 1953.

...

Father Yes, we were talking about flamingos. The point is that the man who wrote Alice was thinking about the same things that we are. And he amused himself with little Alice by imagining a game of croquet that would be all muddle, just absolute muddle. So he said they should use flamingos as mallets because the flamingos would bend their necks so the player wouldn't know even whether his mallet would hit the ball or how it would hit the ball

Daughter Anyhow the ball might walk away of its own accord because it was a hedgehog.

F That's right. So that it's all so muddled that nobody can tell at all what's going to happen.

D And the hoops walked around, too, because they were soldiers.

F That's right—everything could move and nobody could tell how it would move.

D Did everything have to be *alive* so as to make a complete muddle?

F No—he could have made it a muddle by . . . no, I suppose you are right. Because if he'd muddled things any other way, the players could have learned how to deal with the muddling details. I mean, suppose the croquet lawn was bumpy, or the balls were a funny shape, or the heads of the mallets just wobbly instead of being alive, then the people could still learn and the game would only be more difficult—it wouldn't be impossible. But once you bring live things into it, it becomes impossible. I wouldn't have expected that.

...

It would also remain a game if pure noise is injected (eg rolling dice or shuffling cards).

5.6 Schrödinger

ISBN 0-521-42708-8, Canto.

What is Life by Erwin Schrödinger, 1944.

chapters 1-5 make a number of interesting conclusions about DNA before it was discovered.

the $\sqrt{n}$ rule is good to know:

> The laws of physics and physical chemistry are inaccurate within a probably relative error of the order of $1/\sqrt{n}$, where n is the number of molecules that co-operate to bring about that law...

chapter 6 *Order, Disorder and Entropy* is where it gets fun:

> ... Order based on order - Living matter evades the decay to equilibrium - It feeds on 'negative entropy' ... Organization maintained by extracting 'order' from the environment.
>
> What is the characteristic feature of life? When is a piece of matter said to be alive? When it goes on 'doing something', moving, exchanging material with its environoment, and so forth ...

in the epilogue *On Determinism and Free Will* he goes all the way:

> Athman = Brahman ... the mysics of many centuries, independently, yet in perfect harmony with each other ... have described ... the unique experience of his or her life [as] DEUS FACTUS SUM (I have become God) ... their thoughts and joy are *numerically* one ... Consciousnesss is never experienced in the plural ...

see Huxley's *Perenial Philosophy*.

see William James on monotheism vs polytheism in *The Varieties of Religious Experience*.

Mind and Matter, 1958.

he relates consciosness to learning:

> on frequent repetition the whole string of events becomes more and more of a routine, it becomes more and more uninteresting, the resonses become ever more reliable according as they fade from consciousness ... hundreds and hundreds of manipulations and performances of everyday life had all to be learnt once, and that with great attentiveness and painstaking care ... now this whole state of affairs, so well known from the *ontogeny* of our mental life, seems to me to shed light on the *phylogeny* of unconscious nervous processes, as in the heart beat, the peristalsis of the bowels, etc. ... consciousness is associated with the *learning* of the living substance; its *knowing how (Können)* is unconscious.

see Turchin, Zygon.
on Ethics he says:

> Indeed in our days, more perhaps than in others, we hear this demand often enough mocked at 'I am as I am, give room to my individuality! Free development to the desires that nature has planted in me! All the shalls that oppose me in this are nonsense, priests' fraud. God is Nature, and Nature may be credited with having formed me as she wants me to be.' It is not easy to refute their plain and brutal obviousness. ... our conscious life ... is necessarily a continued fight against our primitive ego. ... consciousness and discord with one's own self are inseparable linked up, even that they must be ... proportional to each other.

Hesse *Steppenwolf*

> Dangers to Intellectual Evolution: ... We are ... in grave danger of missing the 'path to perfection' ... the increasing mechanization and 'stupidization' of most manufacturing processes involve the serious danger of a general degeneration of our organ of intelligence. ... by alleviating the responsibility of the individual to look after himself and by levelling the chances of every man, they also tend to rule out the competition of talents and thus to put an efficient brake on biological evolution. ... Next to want, boredom has become the worst scourge of our lives. Instead of letting the ingenious machinery we have invented produce an increasing amount of superfluous luxury, we must plan to develop it so that it takes off human beings all the unintelligent, mechanical, 'machine-like' handling. ... This will not tend to make production cheaper, but those who are engaged in it happier ... there is small hope of putting this through as long as the competition between big firms and concerns all over the world prevails. But this kind of competition is as uninteresting as it is biologically worthless. Our aim should be to reinstate in its place the interesting and intelligent competition of single human beings.

i think he's missing that cultural/technological evolution has taken over and is moving so fast that biology is standing still: eucaryotic cells haven't changed much compared to the organisms made up of them.

the eugenic overtones make me wonder about a connection with the Nazis (he fled when Hitler took over).

see the Unabomber and *New Technology: Society, Employment, and Skill* seem to make similar errors.

The Principle of Objectivation: Neither can the body determine the mind to think, nor the mind determine the body to motion or rest or anything else (if such there be) - Spinoza's Ethics Pt III, Prop 2.

Sir Charles Sherrington's *Man and Nature* ... The Arithmetical Paradox: the Oneness of Mind ... Aziz Nasafi the 13th century Persian mysic ... Plato's theory of forms, the origin of abstraction ... Science and Religion.

5.7 Murray Gell-Mann's *The Quark and the Jaguar*

Murray Gell-Mann's *The Quark and the Jaguar* subtitled 'Adventures in the Simple and the complex', ISBN 0-7167-2725-0, Freeman.

overview of complexity, complex adaptive systems, particle physics, quantum decoherence (i found this section incomprehensible), and biodiversity. liberal doses of dry anecdotes and advertisements for the SFI.

> viii: Those who study complex adaptive systems are beginning to find some general principles that underlie all such systems, ...

pay careful attention to Gell-Mann's intention:

> xiv: [to] dare to take what i call "a crude look at the whole."

ie a high level of abstraction, cf religion.

on p20 he gives a standard evolutionary hierarchy:

his definition of complexity: makes the distinction between the english word, which has a numer of related meanings and senses, only one of which is the mathematical notion of algorithmic complexity (kolmogorov).

but identifies the necessary assumptions (ie sources of subjectivity) as way of course graining (aka abstracting) and choice of language, thus concluding 'any definition of complexity is necessarily context-dependent, even subjective.' (p33).

all-the-same (all zeros) and all-different (all noise) have opposite AIC. but my (the hopefully normal-human) notion of complexity confounds these cases to 'simple' and reserves 'complex' for the living middle ground (visual analogy: the hairy boundary between the inside and outside of the M-set).

Gell-Mann explores this in sections on 'Separating Regularity from Randomness' and 'Randomness vs Complexity', but he moves the problem to the word 'regular' and schema size — essentially compression ratio, but never makes the punch: this abstraction function with its ratio is the result of evolutionary pressure on mind. on p59 he sez:

> A properly operating complex adaptive system would then be unable to find any schema [in noise], since a schema summarizes regularities and there aren't any.

but if one is confronted with a data stream, the size of the schema i find depends on how i think: if maybe i can just memorize it, thus making for a large schema despite the lack of *useful* content. what is useful can: fill my stomach, get me the best mate, avoid predators, etc. it is payoff vs *computational cost*: imagine a species of mind that never forgot anything: it would have to be huge and therefore expensive. this is the information/knowledge/wisdom axis.

p64 gives his definition of life: doesn't abstract it to mean all of complexity. he sticks to a genome, ie passing of information, but he really means 'meaning'.

i wonder if the randomness of QM is a source of part of what makes a good abstraction function: the position of every molecule in a gas is not meaningful because it is chaotically sensitive to quantom randomness. that's a start, but there's much more.

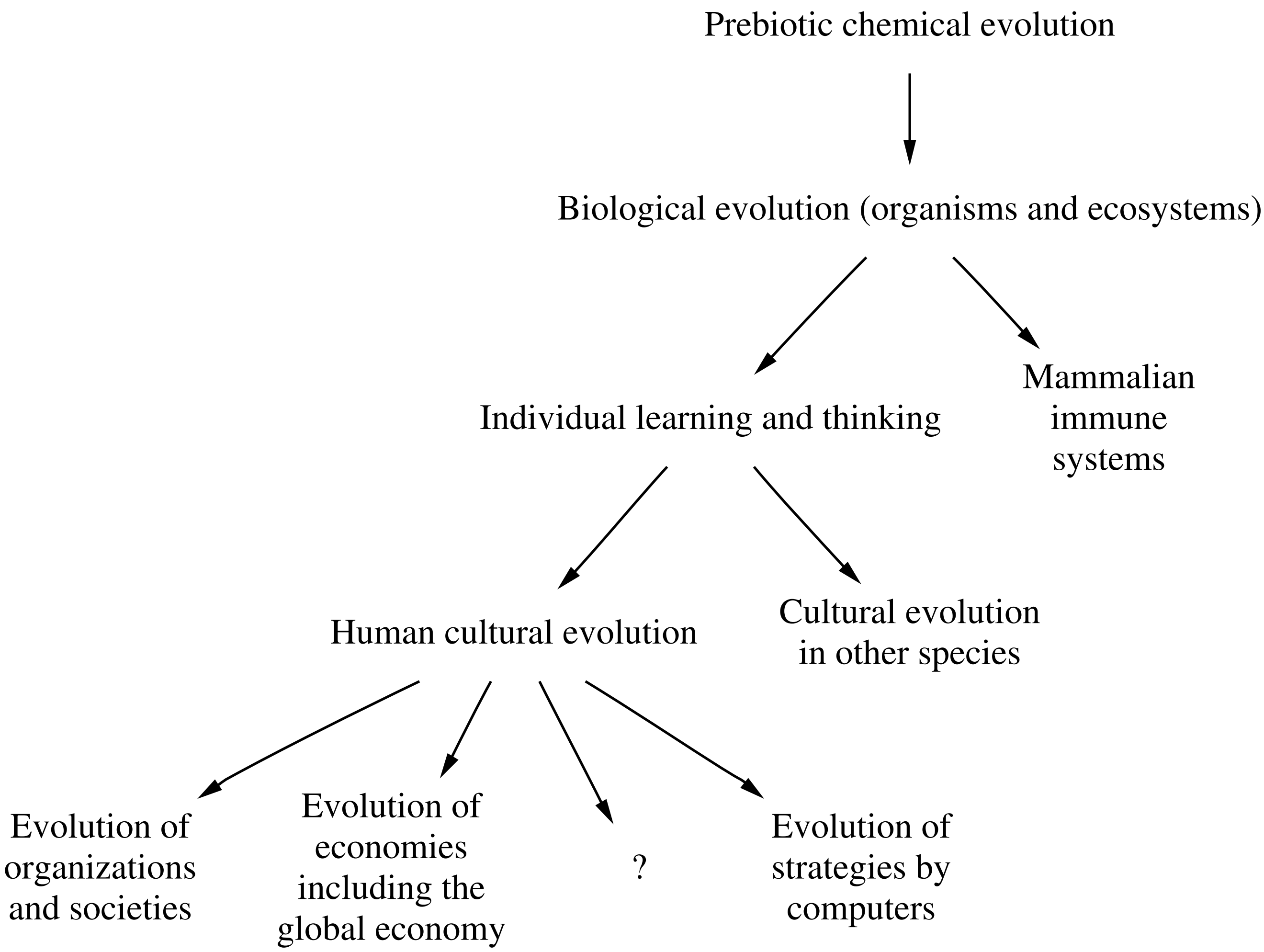

Figure 1: Some complex addaptive systems on Earth.

p94 Zipf's Law, Power Laws, distribution of city populations, distribution of avalanche size in sandpile.

p101

> very roughly, depth is a measure of how laborious is it to go from the brief program or a higly compressed schema to a full-blow description of the system itself or of its regularities? and crypticity measures how laborious is it, starting with the system, to compress its description (or a description of its regularities) into a program or schema

p239 doesn't make a connection between punctuated equilibrium and the power-law of the sandhill.

p244 is there a drive towards high complexity? Gell-Mann is skeptical and cites the increasing distribution of the max of a joint distributions. but he admits there seems to be something going on, but doesn't comment further.

5.8 parameterized graphics

from the perspective of my research, the size of the parameter space is the 'generality' spectrum from power-procedure to shading-procedure to shading-language: the procedure/interpreter axis. in cellular automata, say: life, sum-lut, lut, n-lut, and ga-ca.

life Conway's original, one bit per cell. no parameters.

8sum-life look-up-table (lut) indexed by sum of neighbors and center. conway's is [[000100000][001100000]] (see CAM p20). parameterized by 18 bits.

lut lut indexed by neighbors. parameterized by 2^9 bits.

n-lut handles n-bit cells. parameterized by 2^{9n} bits. practical only for small n.

ga-ca handles n-bit cells (and more). parameterized by recursively defined programs.

the final system, ga-ca uses code sequences instead of luts to save memory, making practical experimentation with larger n (larger parameter spaces). Refer to language of this code as CANL.

most existing programmable CA systems allow the programmer to write the CA rule in a the high-level language CANL (forth is popular). the rule is then executed on all possible neighborhoods to build a lut. in language terms, the rule is compiled into an intermediate code (the lut). (hint: why not compile it to machine code?) (note: the CAM people use special-purpose hardware to implement the lut-interpreter).

why do they do this instead of just calling a CANL interpreter on each cell neighborhood? basically, because it's hard to write a fast data-parallel interpreter. see `proposal/buffering` for more. existing systems either:

- write rules in C and recompiling to try each variation (so the program as the 'user' sees it is not parameterized)
- write rules in CANL and compile to luts, thus limited to low-bit cells

one could also

- write a high-performance CANL interpreter using batching (losing latency and some bandwidth)
- use RTCG to compile CANL programs to machine code

though i don't know of any good examples of either, [Apply] comes close to the latter. building the software tools to support such RTCG is the focus of my thesis research. see `proposal/alternatives.mar`

the succes of this 'inversion' of lut to code depends on which CAs we are interested in (the object-space), and what language the code sequences are 'written' in. this is essentially the hope/assumption of koza-style GP systems: in the coordinate-system of CA space supplied by expression trees, it's 1) easy to express object-space points, and 2) those points are dense enough in terms of the distance metric: the mutation operators.

to explore different languages—to try out new coordinate systems for genetic search—is the motivation for lightweight languages.

5.9 alife bomb

what does alife have to do with bomb? is bomb alive? bomb uses various combinations of images, cellular automata, IFS fractals, but so far no GA. it's parameterized by roughly a dozen integers and several tens of doubles, but it lacks a *recursive* parameter.

working on 'bomb' has basically been a sequence of generalizations. i start with an existing effect, then make some constant into a variable, thus expanding the range of effects and increasing the parameter space. `bomb`.

once my thesis is far enough along i'll add some GA rules to bomb, bumping its 'vitality' up a notch.

5.10 layers

when we talk of emergent behavior, when we talk of interpreters and languages, when we talk of people as made of cells, when we talk of chemistry as above physics, we are talking about layers.

Turchin, one of the originators of MSTT previously did research on *Supercompilation*, a generalization of Partial Evaluation with the same objectives. (see proposal).

as programmers try to provide higher levels of abstraction (more power), programming systems over time usually climb a chain of generalizations. layers are added on top, and intermediate layers (languages) appear.

difference between engineered languages and naturally evolved ones.

we divide the world we see into layers: disciplines with self-consistent rules that we can understand. some of these are only approximations—we've idealized.

5.11 progress

I believe in progress. As a culture we accumulate more and more knowledge, we use science to refine our models of the universe giving us technology: farming, printing, medicine, electricity, telephones, computers, networks.

I'm not saying i think all this stuff is wonderful and good. Just that it exists, and he have to deal with it Brin's The Good and the Bad.

cultures as a whole learn and grow. the economy becomes more efficient and communication improves. each person becomes more capable. the price of computation and communication (information processing) drops. though presumably limited in theory, we still have a long way to go. how far, assuming straight exponential?

I see this as part of a continuing process that begin with life itself, one that will continue to its limits (i think the curve is exponential then levels off, rather than a singularity as predicted by Vinge [link?]) this process is Kauffman's Life force. see the eschaton. see std curve for spread and saturation.

originally people were only connected to those that lived near them; the people they saw daily and weekly. then print hit. then television allowed broadcast to people's largest input chanel: their visual system (videodrome and the new flesh). the internet brought low-bandwidth many-many global connections. more/wider connections result in social creatures. you can start to see the multi-connected video streams in CU-SeeMe.

5.12 the great chain of being

All this talk of layers of complexity defined by learning and modeling sounds a lot like 'the great chain of being', as promoted by Leibniz and Spinoza. I quote Arthur O Lovejoy, *The Great Chain of Being* Lecture V:

> The essential characteristics of the universe are for him plenitude, continuity, and linear gradation. The chain consists of the totality of monads, ranging in the hierarchical sequence from God to the lowest grade of sentient life, no two alike, but each differing from those just below and just above it in the scale by the least possible difference. Since the metaphysics of Leibniz is a form of idealism, or, more precisely, of pan-psychism, the gradation is defined primarily in psychological rather than morphological terms; it is by the levels of consciousness which severally characterize them, the degrees of adequacy and clarity with which they "mirror" or "represent" the rest of the universe, that the monads are differentiated.

so what has happened to these ideas since the 17th century? Schelling added evolution as the means to climb the chain. Since then the theory has fallen out of favor, and the whole thing appears to be regarded as a failure. i have to read more of this book.

5.13 From Complexity to Perplexity

The June 1995 *Scientific American* includes an article *From Complexity to Perplexity* by John Horgan. Subtitled "Can science achieve a unified theory of complex systems? Even at the Sante Fe Institute, some researchers have their doubt" the article is very critical of artificial life and related disciples. I highly recommend it. Below I excerpt some points, and comment on them, [concluding what?]

> The grandest claim of Santa Fe'ers is that they may be able to construct a "unfied theory" of complex systems. John H Holland, a computer scientist with join appointments at the University of Michigan and the Santa Fe Institute, spelled out this breathtakingly ambitions vision in a lecture two years ago: "Many of our most troubling long-range problems–trade balances, unsustainability, AIDS, genetic defects, mental health, computer viruses–center on certain systems of extraordinary complexity. The systems that host these problems–economies, ecologies, immune systems, embryos, neverous systems, computer networks–appear to be as diverse as the problems. Despite appearances, however, the systems do share significant characteristics, so much so that we group them under a single classification at the Santa Fe Insitute, calling them complex adaptive systems [CAS]. This is more than terminology It signals our intuition that there are general principles that govern all CAS behavior, principles that point to ways of solving the attendant problems." Holland, it should be said, is considered to be one of the more modest complexologists.
>
> Some workers now disavow the goal of a unified theory. "I don't even know what that would mean," says Melanie Mitchell, a former student of Holland's who is now at the SFI. "At some level you can say all complex systems are aspects of the same underlying principles, but I don't think that will be very useful." Stripped of this vision of unification, however, the Santa Fe Institute loses much of its luster. It becomes just another place where researchers are using computers and other tools to adress problems in their respective fields. Aren't all scientists doing that?

Building computer models of natural systems is in vogue all over science. An 'applied computer science' where the principles of the implementation and analysis of models of complex systems are studied is useful. To what extent can one perform an experiment with the model instead of the real thing?

How is the field of complexity any different from numerical analysis?

What kind of general principles and unified theories are the immodest talking about? I don't know. If they think they can formalize (compile away) layers of complexity, they are indeed wrong. Murray Gell-Mann, a founder of the SFI makes some claims in his book The Quark and the Jaguar.

> [Herbert Simon says "Most of the people who talk about these great theories have been infected with mathematics,"]

This is one of several swipes at math. I can see the point: the mathematician solves equations and proves theorems. The problem is disposed of and we move onward. It's silly to suggest this will happen to trade balances, mental health, viruses, etc.

> ... the awkward fact that complexity exists, in some murky sense, in the eye of the beholder.

Yes, absolutely. Compexity is relative to an observer (see here). That doesn't mean that it is an unfit target of science—everything is at least a little relative (except maybe math).

> Artificial life–and the entire field of complexity–seems to be based on a seductive syllogism: There are simiple sets of mathematical rules that when followed by a computer give rise to extrememly complicated patterns. The world also contains many extrememly complicated patterns. Conclusion: Simple rules underlie many extrememly complicated phenomena in the world. With the help of powerful computers, scientists can root those rules out.

All phenomena in the world *are* the result of simple rules: the field equations of quantum physics. Sometimes complexity appears in a 'single level' from many applications of simple rules over a simple system, the kind of thing we can model. Such layers are the subject of complexologists. Surely many 'natural systems' of the world are 'big balls of complexity' with no layers and no hope of an efficient model. But the general success of scientific models demonstrates that important systems lie within the scope.

> This syllogism was refuted in a brilliant paper published in *Science* last year. The authors, led by philosopher Naomi Oreskes of Darmouth College, warn that "verification or validation of numerical models of natural systems is impossible." The only propositions that can be verified–that is, proved true–are those concerning "closed" systems, based on pure mathematics and logic. Natural systems are open: our knowledge of them is always partial, approximate, at best.

So what? Who claims otherwise? This repeats the above error of throwing out complexity because it has been 'polluted' with relativism. As if the rest of science stood on firm ground...

> "Like a novel, a model may be convincing –it may ring true if it is consistent with our experience of the natural world," Oreskes and her colleagues state. "But just as we may wonder how much the characters in a novel are drawn from real life and how much is artifice, we might ask the same of a model: How much is based on observation and measurement of accessible phenomena, how much is based on informed judgement, and how much is convenience?"

Do you trust the Turing test? If it quacks...

> [distribution of critical states over time follows a Power law]

I think there's something to this; that is one of the meanings of the word 'pink' in bomb's iconic language.

> ... the [sandpile model with Power behavior] may be so general and so statistical in nature that it cannot really illuminate even those systems it describes. After all, many phenomena can be described by a Gaussian or bell curve.

Yes. And just like there's a theorem behind the bell curve (the Strong Law of Probability), perhaps there's a theorem behind Power law behavior.

> Another skeptic is Philip W. Anderson ... particle physics and indeed all reductionist approaches have only a limited ability to explain the world. Reality has a hierarchical structure, Anderson argued, with each level independent, to some degree, of the levels above and below. "At each stage, entirely new laws, concepts and generalizations are necessary, requiring inspiration and creativity to just as great a degree as in the previous one," Anderson noted. "Psychology is not applied biology, nor is biology applied chemistry."

New laws are only needed because of our limited mental capacity. complex systems are exactly those things that *defy* ability to relate both ends. If there's really anything new at any level, then this is evidence of a god.

One reason that there are more simple systems than you might expect or guess is that a simple system can appear on top of a complex system, *if we use a shoe-horn, idealizing the complex*. Thus weather simulations can work by idealizing the atmosphere, or a population dynamics system by idealizing reproducing individuals, a neural net with Hebbs neurons, or genes, or... Is it really 'simple'? Is it a scientific model? it can't work just right because we've thrown out some information, but it might work well enough to be useful.

> Anderson favors the view of nature described by the evolutionary biologist Stephen Jay Gould of Harvard, who emphasizes that life is shaped less by deterministic laws than by contingent, unpredictable circumstances. [i'm in favor of natural history]

See Gould's Natural History.

5.14 kauffman

> Stuart Kauffman thinks his simulations may lead to the discovery of a "new fundamental force" that counteracts the universal drift towards disorder required by the second law of thermodynamics.
>
> In a book to be published later this year, *At Home in the Universe*, Kauffman asserts that both the origin of life on the earth and its subsequent evolution were not "vastly improbably" but in some fundamental sense inevitable; life, perhaps similar to ours, almost certainly exists elsewhere in the universe.

Yes! I do not believe the life force exists like gravity or magnetism. It's more like statistics, perhaps the Strong Law. Of course Kauffman's 'similar to ours' is meaningless. I'm not too familiar with the term, but 'extropy' sounds about right.

> ... the hope that as computers grow in power, so will science's ability to predict, control and understand nature. Others demurred. Roger N Shepard, a psychologist at Stanford University, worried that even if we can capture nature's intricacies on computers, whose models might themselves be so intricate that they elude human understanding. Francisco Antonio Doria,

> a Brazilian mathematician, smiled ruefully and murmured, "We go from complexity to perplexity." Everyone nodded.

there's no doubt that more powerful computers will allow better modeling and the benefits thereof. Even though we may never 'understand' one of our models—because it's complex beyond our minds, its still useful! Our perplexity is really wonderment that 'that's all there is to it'.

5.15 Gould's Natural History

from Stephen Jay Gould's book *Wonderful Life*. ISBN 0-393-30700-X, WW Norton.

summary: tells the story of the precambrian explosion and the Burgess Shale in three ways:

- a piece of the history of science
- a lesson in paleontology, biology, and anatomy
- a philisophical/political/moral story: a debunking of evolution as a *linear march of progress* and a *cone of increasing diversity* — and their anthropocentrism and racist connections.

it is the last of these that concerns us here.

for example on p28

> The familiar iconographies of evolution are all directed—sometimes crudely, sometimes subtly—toward reinforcing a comfortable iew of human inevitability and superiority. The starkest version, the chain of being or ladder of linear progress, has an ancient, pre-evolutionary pedigree (see A. O Lovejoy's classic, *The Great Chain of Being*, 1936) Consider for example, Alexander Pope's *Essay on Man*
>
> > Far as creation's ample range extends
> > The scale of sensual, mental powers ascends:
> > Mark how it mounts, to man's imperial race,
> > From the green myriads in the peopled grass.
>
> p43 ... this error infests technical as well as lay discourse, an editorial in *Science* constructs an order every bit as motley and senseless as White's "regular gradation". Commenting on species commonly used for laboratry work , the editors discuss the "middle range" between unicellular creatures and guess who at the apex: "higher on the evolutionary ladder," we learn, "the nematode, the fly, and the frog have the advantage of complexity beyond the single cell, but represent far simpler species than mammals"

Gould justifies 'Contingency' in natural history. His prime thought experiment is "replay the tape, and things would turn out completely different". The main content of the book presents evidence from the fossils of the Burgess shale. He attacks the notion of evolution as progress towards perfection, and placing *homo sapiens* at the apex.

Gould asserts that those few phyla that survived the shale to form life as we know it today were probably ‘lucky’ rather than ‘more fit’: in an environmental catastrophe survival is a lottery.

Later on Gould adds without argument or elaboration “of of the thousands [of possible histories] only a few contain anything like self consciousness”. Why? it seems to me that flops (aka smarts, information processing/communication) are in the limit very cheap, but have a large advantage (especially in adaptability and handling new and different situations—eg rapid environmental change. exactly what may cause decimation). the result would be alien intelligence.

6 eschaton

The eschaton is the end of the world as we know it. while the word itself is very old [according to the OED, eschatology is the the department of theological science concerned with the four last things: death, judgement, heaven, and hell] it has acquired a recent connotation: the end of the world brought on by increasing technology, communication, and connectedness: a singularity or omega point.

Terence McKenna has this to say (original here)

> An interesting thing about this concatenation of connectedness is that each stage of its condensation took place more rapidly than the stages that preceded it. At its birth the Universe was as a pure plasma, there were no atomic systems; there was so much energy within the system that electrons either did not yet exist or were unable to settle into stable orbits. Then, as the Universe cooled, atomic systems began to form; stars condensed and through nuclear chemistry cooked up heavier elements, from which eventually developed a carbon based chemistry. This opened the possibility of molecular chemistry-new realms of connectedness-a new proliferation of opportunity for novelty. This opportunity led to life, to higher animals, to culture and eventually and comparatively recently to culture and epigenetic coding systems such as language and even more recently writing. The legacy of the conservation of connectedness is the meta-connected chaostrophy of Twentieth Century planetary culture. My model-building has sought to unify all of these diverse phenomena and to treat them as manifestations of a single set of laws; laws which describe the ingression of novelty into time And its conservation and concentration in ordinary space/time and ordinary immediate experience.

he uses Alfred North Whitehead’s definition of novelty (original here):

> Creativity is the principle of novelty. Creativity introduces novelty into the content of the many, which are the universe disjunctively. The creative advance is the application of this ultimate principle of creativity to each novel situation which it originates. The ultimate metaphysical principle is the advance from disjunction to conjunction, creating a novel entity other than the entities given in disjunction. The novel entity is at once the togetherness of

> the 'many' which it finds and also it is one among the disjunctive ' many' which it leaves; it is a novel entity, disjunctively among the many entities which it synthesises. The many become one, and are increased by one. In their natures, entities are disjunctively 'many' in process of passage into conjunctive unity... Thus the 'production of novel togetherness' is the ultimate notion embodied in the term concrescence. These ultimate notions of 'production of novelty' and 'concrete togetherness' are inexplicable either in terms of higher universals or in terms of the components participating in the concrescence. The analysis of the components abstracts from the concrescence. The sole appeal is to intuition. (Process and Reality, p. 26)

Did you understand that? I'm still thinking about it. It sounds like he's talking about emergence more than anything else. Maybe I should read the book...

Though I think McKenna is crazy in many ways I think the basic idea of increasingly increasing novelty/life/complexity is correct. The fundamental question: what is the topology of the curve? Some predict it becomes infinite in finite time (in 2012 in McKenna's case), this is a singularity. But this injection of the infinite is a religious assertion, just like Tipler's Omega point.

Feigenbaum and Jumping Jesus.

I see no reason for this. To me, the curve appears exponential. This has the important property of scale-neutrality: no matter where you are on the curve (ie no matter what age you live in) it looks the same. Everyone always thinks they live on the verge of great change, because relative to their past, they are! Until we approach the physical limits of computation and energy (if there are any) I see no reason for this to change.

Hans Moravec has some cogent words about the future, see Pigs in Cyberspace and these excerpts from *The Age of Mind: Transcending the Human Condition through Robots*: one and two.

6.1 what is god?

what is abstract/essense/god?

god = faith = assumptions = axioms

life force, spirit, ghost in the machine.

the infinite, induction, axiom of choice.

fixing the point, solving the recursive equation, closing the loop.

science as religion, occam's razor.

objectivity of mathematics.