

# What Use Is Half An Algorithm?

Simon Hickinbotham

Simon Hickinbotham: YCCSA, University of York, Heslington, York YO1 5DD, UK  
email:sjh@cs.york.ac.uk

9 March, 2011

# Abstract

You may have heard of the phrase "what use is half an eye?", which is often used in an attempt to refute biological evolution. The answer that evolution provides to this question is that half an eye is precisely 1% more use than 49% of an eye, provided appropriate intermediate states exist between having 100% of an eye or no eye at all (Dawkins). This process allows the development of sophisticated visual apparatus without the need for a designer. If we want to develop new algorithms using evolution, we must provide a framework in which it is possible for an algorithm to be measurably 49% competent and we must be capable of producing further algorithms that have a range of competencies around the 49% mark. If we are writing an algorithm for some optimisation problem, then it is quite easy to see that it is possible to be measurably 49% competent. There are two issues here. Firstly, we have to be able to cast our problem as a problem of optimisation. Secondly, we have to provide an appropriate solution space in which the competence can vary from the start, and in which an acceptable solution must exist. Computer scientists seek to use evolution to develop sophisticated solutions to problems in just about any application that computers can be used for. Thus we must ask the question "What use is half an algorithm?". In this talk, I will consider the ramifications of this question, and present work we have been doing in the Plazzmid project to address the issues that have been raised.

1. Part 1: How we came to ask the question
2. Part 2: How we've gone about finding an answer

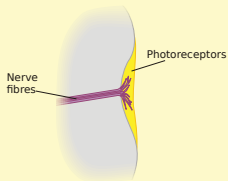
# Motivation

- ▶ Designing stuff is hard! and it is getting more difficult as the complexity of the engineered world increases
- ▶ We would like to look at automating this process as much as possible - free the process from the limitations of human ingenuity.
- ▶ We'd also like to be able to throw stuff together, and stand back as the system integrates itself.
- ▶ Evolution is a potential way of doing this

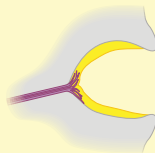
## Why ask “What use is half an algorithm?”..?

- ▶ (Biol) Evolution wrestles with the idea of (intelligent) design
- ▶ This can be caricatured by the question “what use is half an eye?”
- ▶ The answers to the “eye” question have implications.
- ▶ Let’s look at this first...

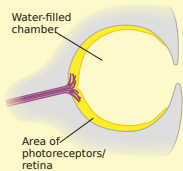
a) Region of photosensitive cells



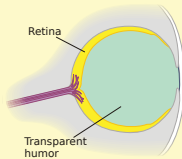
b) Depressed/folded area allows limited directional sensitivity



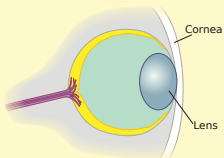
c) "Pinhole" eye allows finer directional sensitivity and limited imaging



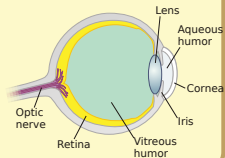
d) Transparent humor develops in enclosed chamber



e) Distinct lens develops

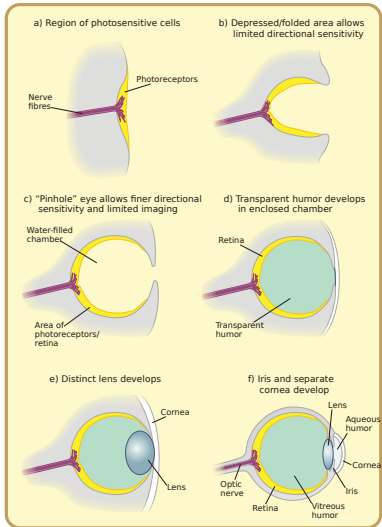


f) Iris and separate cornea develop

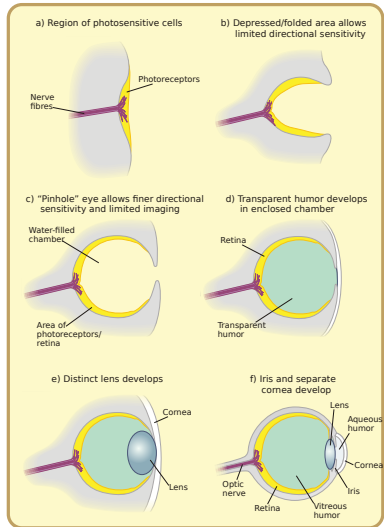


## Nick Lane “Darwin and the eye”

- ▶ Darwin (often quoted): “to suppose that the eye evolved by natural selection seems, I freely confess, absurd in the highest possible degree.”
- ▶ Darwin (rarely quoted): “Though insuperable to our imagination, the difficulty can hardly be considered real.”
- ▶ Darwin argued that the eye could evolve, so long as the three conditions of natural selection are met.

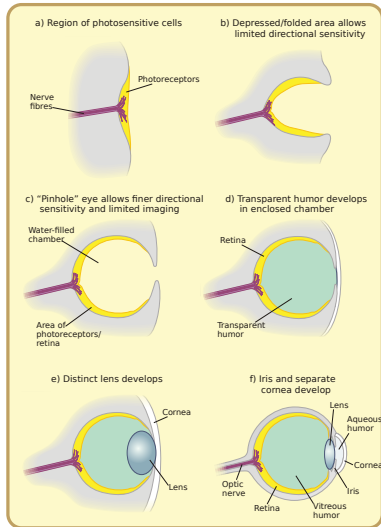


- ▶ First, eyesight should **vary** between individuals.
- ▶ Second, these variations should be *heritable*.
- ▶ Third, they should make some difference to *survival*.



- ▶ One way to think about the behaviour of these eyes is to consider Dennet's stances:
  1. *Physical stance*: The physical properties that endow the system with capability
  2. *Design stance*: Purpose, function and design
  3. *Intentional stance*: The specification that led to the design

- ▶ The function of the eye is to:
  - ▶ *Physical stance*: allow focussed light to be passed into an array of neurons
  - ▶ *Design stance*: to perceive the external world
  - ▶ *Intentional stance*: to make an informed decision about successful survival

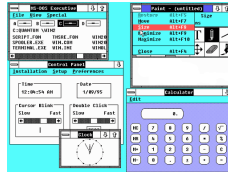


- ▶ “What use is half an eye?”
  - ▶ A “design stance” question about an evolved system that is easily rebutted.
  - ▶ The “intentional stance” view of the eye provides the answer - survival
- ▶ “What use is half an algorithm?”
  - ▶ An “intentional stance” question about an engineered system.
  - ▶ How do we start to think about this?

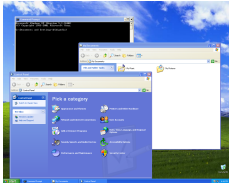
Now let's look at the development of a well-known engineered system:



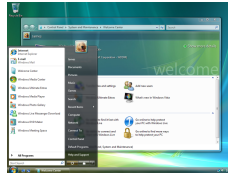
Windows 1.0



Windows 2.0



Windows XP



Windows Vista

"Microsoft didnt make any big promises about application compatibility, and its a damn good thing. If a desktop application didnt follow Vistas rules for behaviour, Vista wouldnt let it run. The program would fail to load, crash on use, or eat the users data, depending on the development infraction. And to be clear, were not talking about shareware apps created by some dude in his basement, were talking about Acrobat Reader, iTunes, Trillian, and dozens of other programs, not even counting the anti-virus programs that are rarely compatible with a new OS."

- ▶ We can see that engineered systems can be brittle, and we spend a great deal of time, money and creativity in maintaining these systems on a narrow track.
- ▶ But to get an evolutionary answer to this, we have to have *variability, heredity and selection*
- ▶ So how might we begin to introduce some variability into these systems without crashing them?
- ▶ Here we take some inspiration from the “physical stance” view of evolution, and build up a new “intentional stance”



EPSRC grant no. EP/F031033/1

## Plazzmid goals in one slide

- ▶ Bacteria share genetic info via transfer of plasmid
- ▶ transposons move genes around the genome
- ▶ transposons encode the function of transposition
- ▶ beneficial genes occasionally hitch a ride on transposons
- ▶ The idea is that the genome is arranged to exploit this via evolution

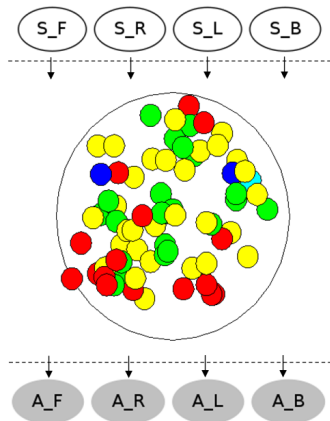
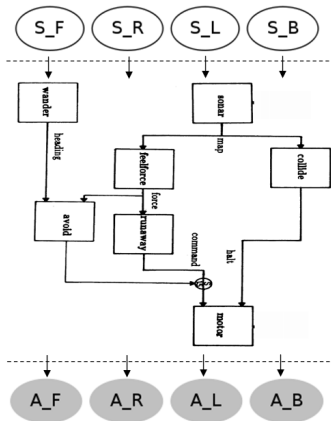
# Evolving genome-curating machinery

- ▶ We can construct *abstract* networks of some of these ideas
- ▶ But the problem is that we want the machinery to evolve.
- ▶ Recall the three components we need:
  1. **Variation**
  2. heredity
  3. selection.
- ▶ “Building” variation into embodied is hard for two reasons:
  1. The machines have to be *embodied*, or the variation is fake.
  2. Maintaining function in the face of variation is difficult.
- ▶ **Thus the question: “what use is half an algorithm”..**

# Physical stance view of biology

- ▶ Let's consider how molecules interact in a biological system.
- ▶ Firstly, there are almost always very many instances of any protein. These proteins have both collective and individual variation
- ▶ Their behaviour is massively parallel
- ▶ They can only bring about collapse of the system by triggering cascades of reaction – a single interaction cannot cause a global crash directly.

# Conceptual comparison



- ▶ crisp logic and crisp routing are good for engineering, but require evolution to take large leaps in design space.
- ▶ fuzziness of a system could be useful in some way - soften the changes made
- ▶ We want to consider a possible interaction between any component in a system with any other.
- ▶ functional plasticity is important too.

# Motivating an artificial chemistry

- ▶ We want soft interactions and soft execution
- ▶ We want it cheaply, and it must never break the system.
- ▶ Molecular dynamics are too costly. Lets break it down into binding and execution of molecular "programs" instead.
- ▶ This has been done before - Avida for example. To use these ideas, we must:
  1. make binding "soft"
  2. get rid of registers, memory etc - encode all this on the sequence if we need it.
  3. make execution "soft" - use the same mech. as for binding if possible.
  4. And lets specify that only two things can interact - keep it simple.

## Why the emphasis on soft binding?

- ▶ Basis of all interesting DNA interactions
- ▶ Transposons, promoters etc. *all* use specific binding
- ▶ Therefore *necessary* for phenotypic control of gene expression

OBE**Q**X**U**T**U**D**Y**GRHB



*Complement*

BOR**F**OK**H**GH**Q**L**T**EUO



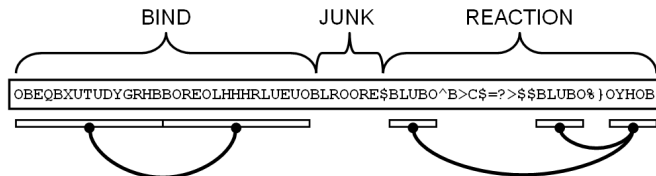
*Inexact alignment*

BOREOLHHHRLUEUO

# “soft” binding with Smith-Waterman algorithm

$s'$	$C(s')$	A	A	A	B	H	G	>	\$	C	D	G	B	A	A	A	
N	A	0	1.00	0	0	0	0	0	0	0	0	0	0	0	0	0	
O	B	0	0	0.88	0.88	2.00	0.67	0	0	0	0	0	0	1.00	0	0.88	
N	A	0	<u>1.00</u>	1.00	1.88	0.76	0.88	0	0	0	0	0	0	2.00	1.00	1.88	
O	B	0	0	<u>0.88</u>	0.88	2.88	1.55	0.22	0	0	0	0	1.00	0.67	1.88	0.88	
N	A	0	1.00	1.00	<u>1.88</u>	1.55	1.76	0.55	0	0	0	0	0	2.00	1.67	2.88	
O	B	0	0	0.88	<u>0.88</u>	<u>2.88</u>	1.55	0.88	0	0	0	0	1.00	0.67	1.88	1.55	
U	H	0	0	0	0	<u>1.55</u>	<u>3.88</u>	2.55	1.22	0	0	0	0	0	0.55	0.76	
T	G	0	0	0	0	<u>0.22</u>	<u>2.55</u>	<u>4.88</u>	<u>3.55</u>	2.22	0.89	0	1.00	0	0	0	
\$	\$	0	0	0	0	0	1.22	3.55	4.38	<u>4.05</u>	2.72	1.39	0.03	0.75	0	0	
P	C	0	0	0	0	0	0	2.22	3.05	4.26	<u>5.05</u>	3.72	2.39	1.06	0.50	0	
Q	D	0	0	0	0	0	0	0.89	1.72	2.93	4.14	<u>6.05</u>	<u>4.72</u>	3.39	2.06	0.73	
O	B	0	0	0	0	0	0	0	0.39	1.60	2.81	4.72	<u>5.72</u>	4.39	3.06	1.73	
N	A	0	1.00	1.00	1.00	0	0	0	0	0.27	1.48	3.39	3.84	5.05	<u>6.72</u>	5.39	
N	A	0	1.00	2.00	2.00	0.88	0	0	0	0	0.15	2.06	2.51	3.72	6.05	<u>7.72</u>	
P	C	0	0	0.75	1.75	1.88	0	0	0	0	1.00	0.73	1.31	2.39	4.72	6.39	7.47
N	A	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	
O	B	.	/	/	/	/	/	/	/	/	/	/	/	/	/	/	
O	B	.	/	/	/	/	/	/	/	/	/	/	/	/	/	/	
N	A	.	.	/	/	/	/	/	/	/	/	/	/	/	/	/	
O	B	.	/	/	/	/	/	/	/	/	/	/	/	/	/	/	
U	H	.	/	/	/	/	/	/	/	/	/	/	/	/	/	/	
T	G	.	/	/	/	/	/	/	/	/	/	/	/	/	/	/	
\$	\$	.	/	/	/	/	/	/	/	/	/	/	/	/	/	/	
P	C	.	/	/	/	/	/	/	/	/	/	/	/	/	/	/	
Q	D	.	/	/	/	/	/	/	/	/	/	/	/	/	/	/	
O	B	.	/	/	/	/	/	/	/	/	/	/	/	/	/	/	
N	A	.	/	/	/	/	/	/	/	/	/	/	/	/	/	/	
N	A	.	/	/	/	/	/	/	/	/	/	/	/	/	/	/	
P	C	.	/	/	/	/	/	/	/	/	/	/	/	/	/	/	

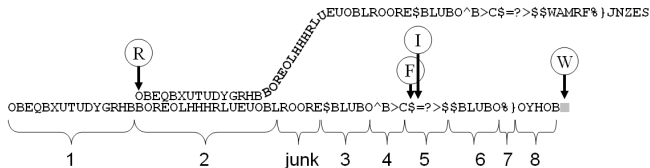
# Stringmol replicase molecule



- ▶ Note that replicases were the initial target function for the Stringmol “language” - we wanted a system that survived in the face of constant decay by a process of self-copying.
- ▶ Here’s a replicase stringmol. Note there are *regions* to the sequence that are *embodiments* of what the program does at that stage.

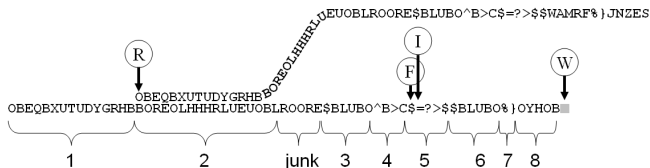
# Binding molecules to form virtual machines

- ▶ Bind location determines:
  - ▶ which molecule is “active”
  - ▶ where the reaction-program commences (positions pointers)
- ▶ The strings specify the program:
  - ▶ Templates  $T = \{ \text{'A'}, \dots, \text{'Z'} \}$
  - ▶ Functional:  $\Phi = \{ \text{'\$'}, \text{'>'}, \text{'^'}, \text{'?'}, \text{'='}, \text{'\%'}, \text{'\}' } \}$
- ▶ Four pointer types: **I**nstruction; **F**low; **R**ead; **W**rite



# Mutation On Copy

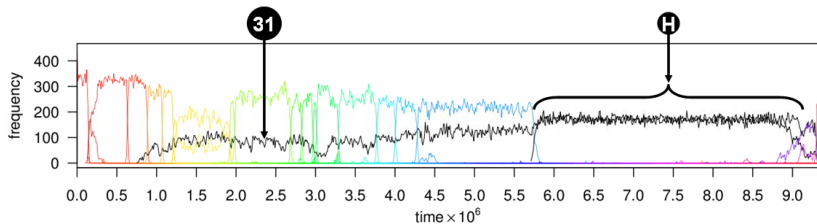
- ▶ The '=' instruction copies from the Read pointer to the Write pointer
- ▶ This has a small chance of *error* ( $p = 0.0001$ )
- ▶ mutation to “next door” symbols on a pre-arranged sequence
- ▶ Mutation rate *per molecule* is a function of string length
- ▶ Now we have turned mutation “on”, what will happen to a replicase system? Will it survive? Will it change?



# Setting up a single trial

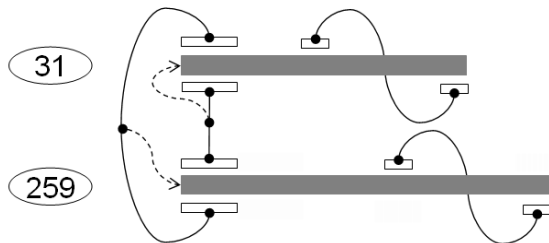
- ▶ 400 identical “seed” replicases
- ▶ Stochastic chemical simulation
- ▶ Limited energy per time-step
- ▶ Constant decay rate for all molecules
- ▶ Survival via (inexact) copying
- ▶ Run until no stringmols remain in the system

# Observations of an individual trial



- ▶ **31**: persists for  $9 \times 10^6$  time steps
- ▶ **H**: Hypercycle emerges
- ▶ Partners in this hypercycle do not self-copy

# Reactions in hypercycle partners



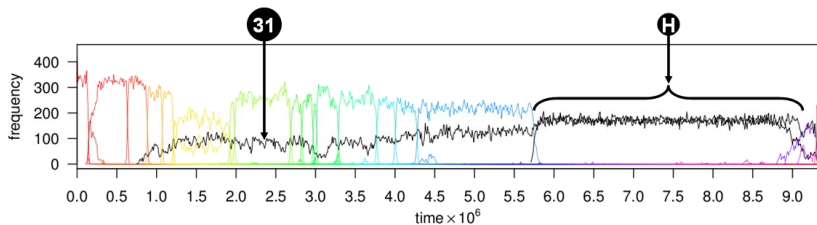
# Origin of the macro mutation

- ▶ A mutation in the functional region causes a double-length molecule to be created

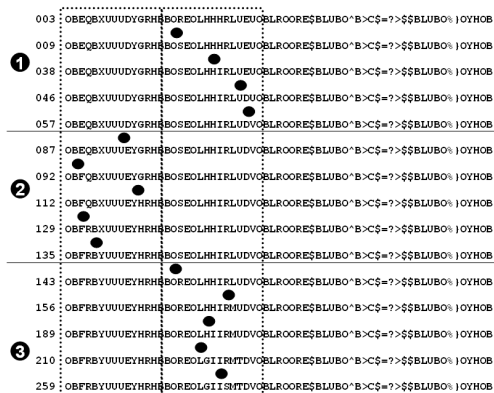
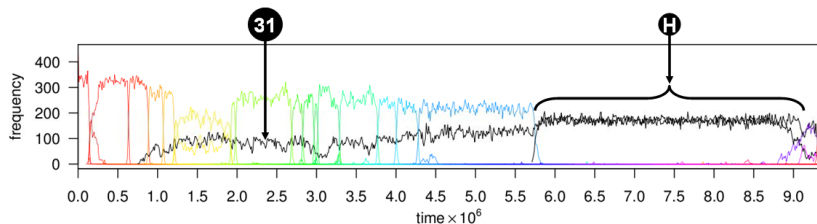
```
030 OBEQB XUUDYG...P^B>C$=?>$$BLUBO%}OYHO OBEQB XUUDYGRHBB OSEOLHHHRLUEUOBLROORE$BLUBO^B>C$=?>$$
Bind site:          |-----|
009                OBEQB XUUDYGRHBB OSEOLHHHRLUEUOBLROORE$BLUBO^B>C$=?>$$BLUBO%}OYHOB
Product:           |-----|
031                BBOSEOLHHHRLUEUOBLROORE$BLUBO^B>C$=?>$$BLUBO%}OYHOB
```

- ▶ Longer alignment with centre of species 030.
- ▶ Species 009 is the template, species 030 is active
- ▶ First binding site on 009 is not copied - species 031 is created

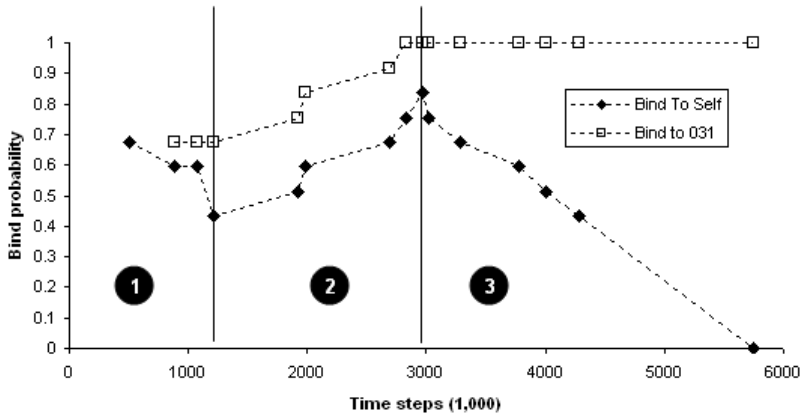
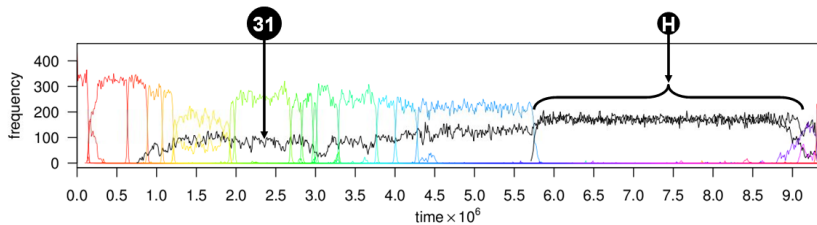
# Repeated mutation $\rightarrow$ rich behaviour



# Tracking the dominant mutations...



# Evolution of binding



# Stringmol NOT gate

- ▶ We've tailor made an AC to experiment with an embodied replicase system
- ▶ Can we use the language more generally?
- ▶ Let's see if we can construct a NOT gate

For a “true” signal, we specify:

```
| -0- | | -X- |  
INPUTTTTTT
```

and for false:

```
| -0- | | -X- |  
INPUTFFFFF
```

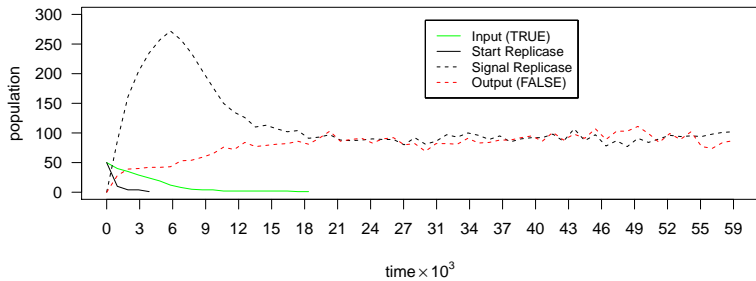
# Stringmol NOT function:

```
|-----Replicase-----|
|---1---||3'|--1'---|      |-2-|          |-2-| |-4'| |-7-|
EEEEEEEEEXYZRRRRRRRRRABC$BLUBO^B>C$=?>$BLUBO%BXJTY}$PBCPY>

|Bind-input--||-Mod-replicase-||-Check-boolean--|
|-0'|      |-3-|  |-4-|      |0'|  |-5-|
VACHG^B?VACH}$KLMEE>C=$OKWGL>C=^$CHG>B^$LHVBC?GG>

|--Set-output-false-----||-Set-output-true-----|
|-6-|  |-7-|  |-6'|      |-5'|  |-8-|  |-7-|  |-8'|
$SNYFR>B$PBCPY>XFALSEFFFFF}$YUIOPX$YYEHR>B$PBCPY>LLRUETTTT}$

|-Make-output-message-----||-Express-output-message-----|
|-7'|  |9|      |10'|  |9'|      |10-|  |-2-|          |-2-| |-2'|
COPCLXX$XM>C=====BOZDEODTWKZQQQQQ$OBMQR>B$BLUBO>C=====BLUBO%OYHOB
```



# Summary

- ▶ Half an algorithm *could be useful* if we take the intentional stance.
- ▶ Novel architectures **could** be generated through evolutionary processes *if they were designed with this in mind*
- ▶ Stochastic mixing, soft handshaking, and functional plasticity are all needed to achieve this.
- ▶ But these systems are hard to bootstrap because we are used to designing with crisp algorithms
- ▶ we need to identify appropriate *functional domains* in ever more suitable languages if we want to pull this off.

# What Use Is Half An Algorithm?

Simon Hickinbotham

Simon Hickinbotham: YCCSA, University of York, Heslington, York YO1 5DD, UK  
email:sjh@cs.york.ac.uk

9 March, 2011