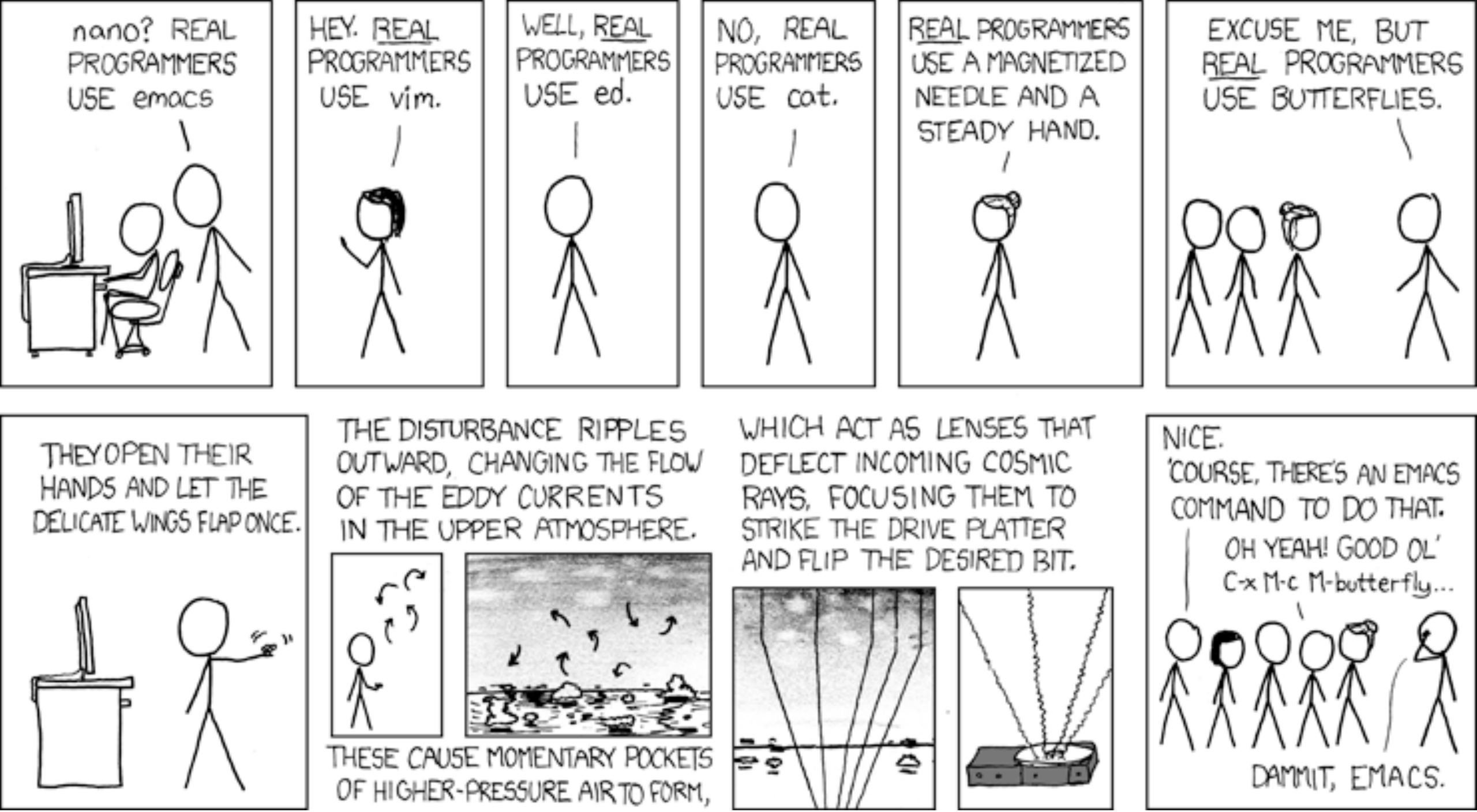

nano? REAL PROGRAMMERS USE emacs
HEY. REAL PROGRAMMERS USE vim.
WELL, REAL PROGRAMMERS USE ed.
NO, REAL PROGRAMMERS USE cat.
REAL PROGRAMMERS USE A MAGNETIZED NEEDLE AND A STEADY HAND.
EXCUSE ME, BUT REAL PROGRAMMERS USE BUTTERFLIES.
THEY OPEN THEIR HANDS AND LET THE DELICATE WINGS FLAP ONCE.
THE DISTURBANCE RIPPLES OUTWARD, CHANGING THE FLOW OF THE EDDY CURRENTS IN THE UPPER ATMOSPHERE.
THESE CAUSE MOMENTARY POCKETS OF HIGHER-PRESSURE AIR TO FORM,
WHICH ACT AS LENSES THAT DEFLECT INCOMING COSMIC RAYS, FOCUSING THEM TO STRIKE THE DRIVE PLATTER AND FLIP THE DESIRED BIT.
NICE.
'COURSE, THERE'S AN EMACS COMMAND TO DO THAT.
OH YEAH! GOOD OL' C-x M-c M-butterfly...
DAMMIT, EMACS.

xkcd.com

# Logistics

- We plan to return graded projects on ~~Wednesday,~~ Thursday.
- I will hold office hours all day Thursday so you can pick up and discuss your graded project.
- Project 2 has been posted.
- Groups have been assigned.

# Logistics

- We plan to return graded projects on ~~Wednesday,~~ Thursday.
- I will hold office hours all day Thursday so you can pick up and discuss your graded project.
- Project 2 has been posted.
- Groups have been assigned.

**¿Project Questions?**

# Genetic Algorithms Continued

Lecture 10

# Terminology

On chromosomes:

4 Bases {T,C,A,G}

Codons (sequences of 3 bases) code for each amino acid.

Genes (variable length sequences of codons) code for complete proteins.

Proteins form the Phenotype.

| Amino acids biochemical properties | nonpolar | polar | basic | acidic |
|---|---|---|---|---|

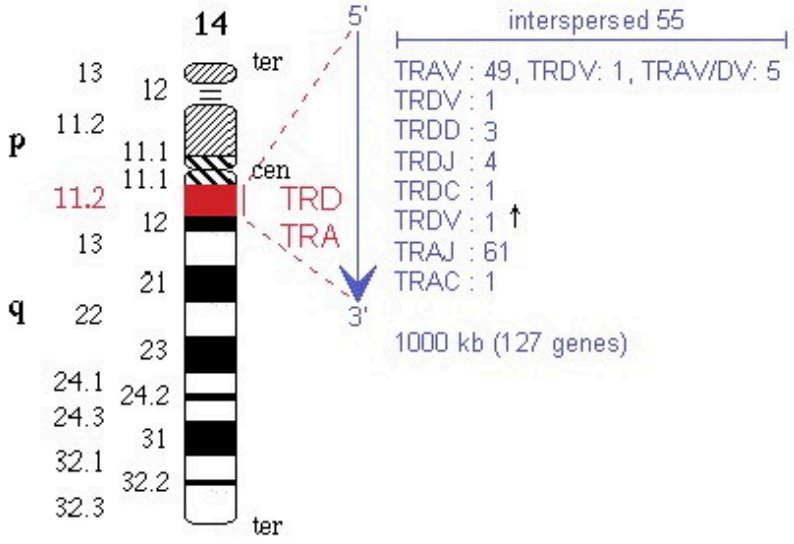
| Termination: stop codon |
|---|

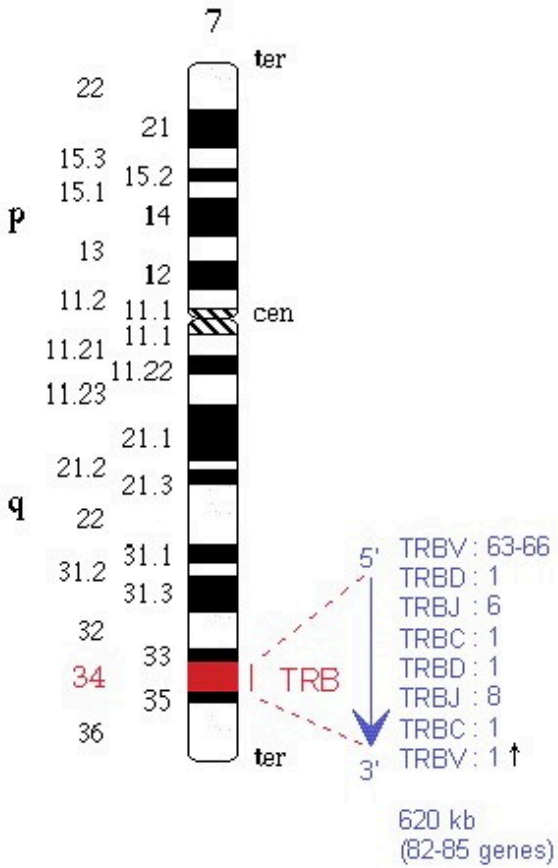
**Standard genetic code**

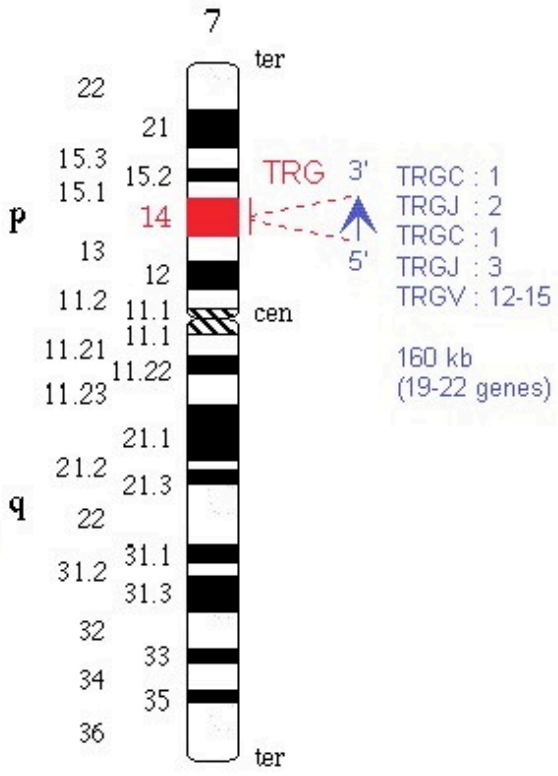
| 1st base | 2nd base: T | | 2nd base: C | | 2nd base: A | | 2nd base: G | | 3rd base |
|---|---|---|---|---|---|---|---|---|---|
| T | TTT | (Phe/F) Phenylalanine | TCT | (Ser/S) Serine | TAT | (Tyr/Y) Tyrosine | TGT | (Cys/C) Cysteine | T |
| | TTC | | TCC | | TAC | | TGC | | C |
| | TTA | (Leu/L) Leucine | TCA | | TAA[B] | Stop (*Ochre*) | TGA[B] | Stop (*Opal*) | A |
| | TTG | | TCG | | TAG[B] | Stop (*Amber*) | TGG | (Trp/W) Tryptophan | G |
| C | CTT | | CCT | (Pro/P) Proline | CAT | (His/H) Histidine | CGT | (Arg/R) Arginine | T |
| | CTC | | CCC | | CAC | | CGC | | C |
| | CTA | | CCA | | CAA | (Gln/Q) Glutamine | CGA | | A |
| | CTG | | CCG | | CAG | | CGG | | G |
| A | ATT | (Ile/I) Isoleucine | ACT | (Thr/T) Threonine | AAT | (Asn/N) Asparagine | AGT | (Ser/S) Serine | T |
| | ATC | | ACC | | AAC | | AGC | | C |
| | ATA | | ACA | | AAA | (Lys/K) Lysine | AGA | (Arg/R) Arginine | A |
| | ATG[A] | (Met/M) Methionine | ACG | | AAG | | AGG | | G |
| G | GTT | (Val/V) Valine | GCT | (Ala/A) Alanine | GAT | (Asp/D) Aspartic acid | GGT | (Gly/G) Glycine | T |
| | GTC | | GCC | | GAC | | GGC | | C |
| | GTA | | GCA | | GAA | (Glu/E) Glutamic acid | GGA | | A |
| | GTG | | GCG | | GAG | | GGG | | G |

Wikipedia, 2-22-2017

# Chromosomal location of the TCR α/δ, β and γ chain loci in man




locus representations from the International Immunogenetics Information System (IMGT) server (http://www.imgt.org/)

TCR: T cell Receptor

These loci are highly variable in their alleles.

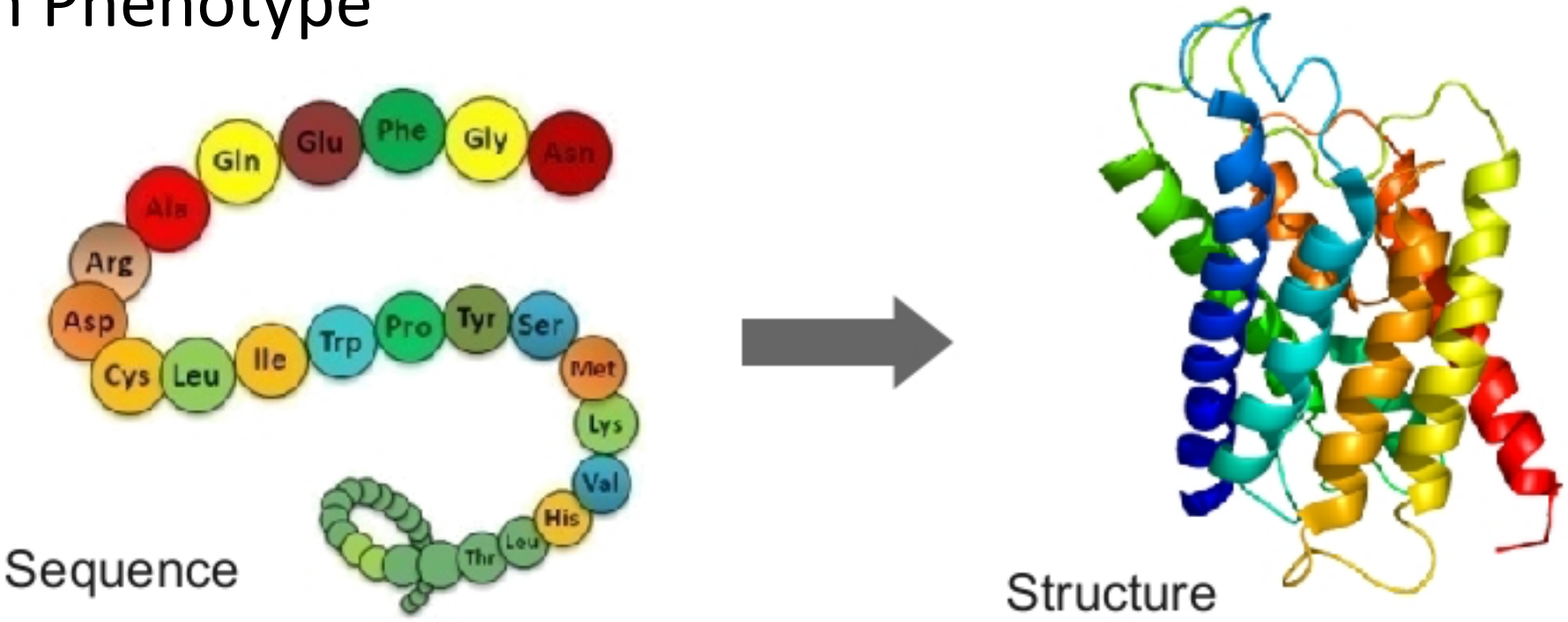
# Terminology: Protein Phenotype

Proteins form the phenotype.

The conformation of a protein is how it twists itself.

This conformation process is dynamical.

Proteins vibrate into low energy conformations.






Robot path planning techniques for protein folding. (Lydia Tapia, UNM)

Xinyu Tang, Shawna Thomas, Lydia Tapia, David P. Giedroc, Nancy M. Amato, "Simulating RNA Folding Kinetics on Approximated Energy Landscapes," *Journal of Molecular Biology*, 3811(4):1055-1067, 2007

# Terminology: Protein Phenotype

Proteins form the phenotype.

The conformation of a protein is how it twists itself.

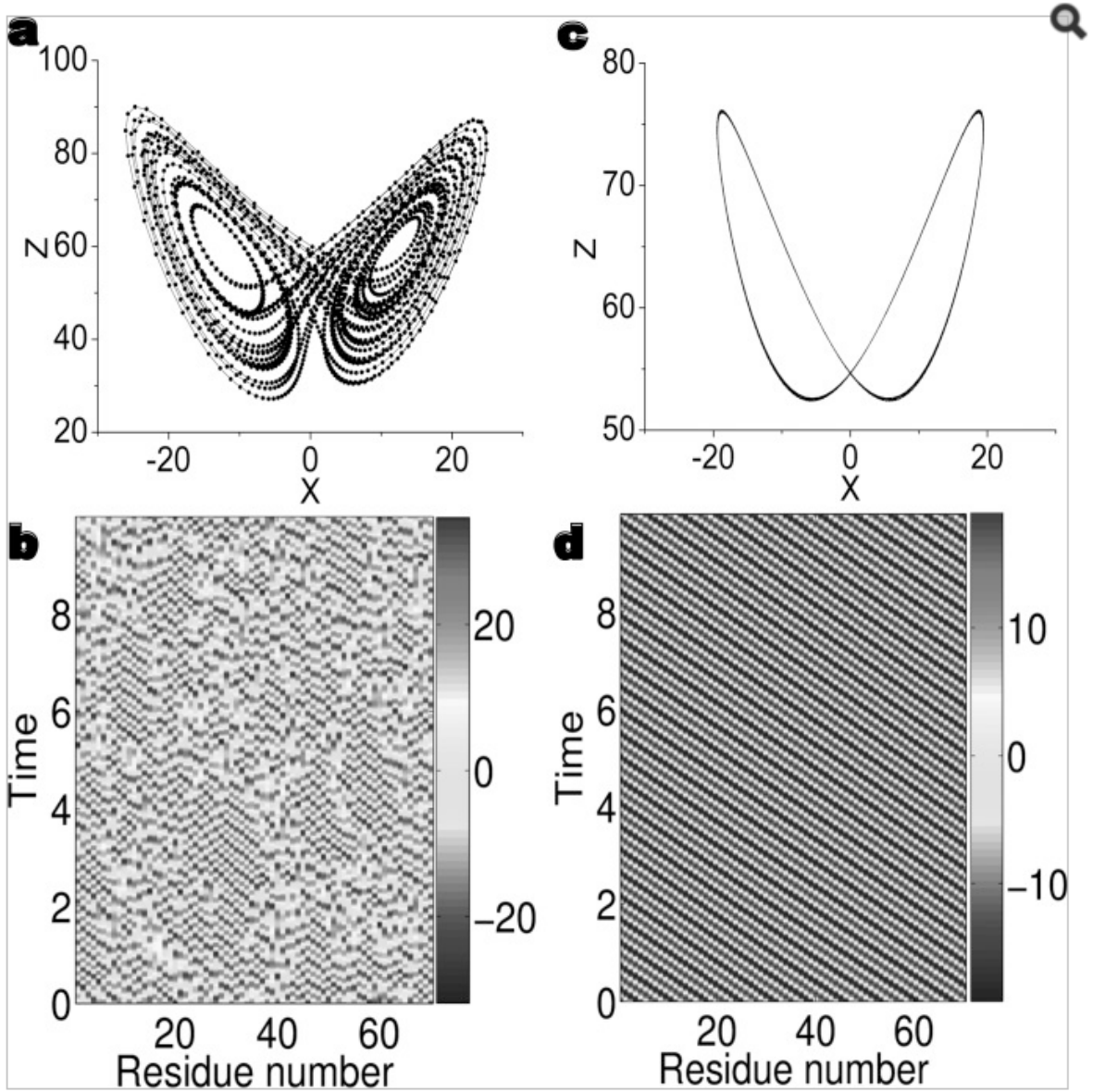
This conformation process is dynamical.

Proteins vibrate into low energy conformations.

These states are defined by stable and unstable fixed points and saddle points.

Xia, Kelin, and Guo-Wei Wei. "Molecular nonlinear dynamics and protein thermal uncertainty quantification." *Chaos: An Interdisciplinary Journal of Nonlinear Science* 24.1, 013103, 2014.

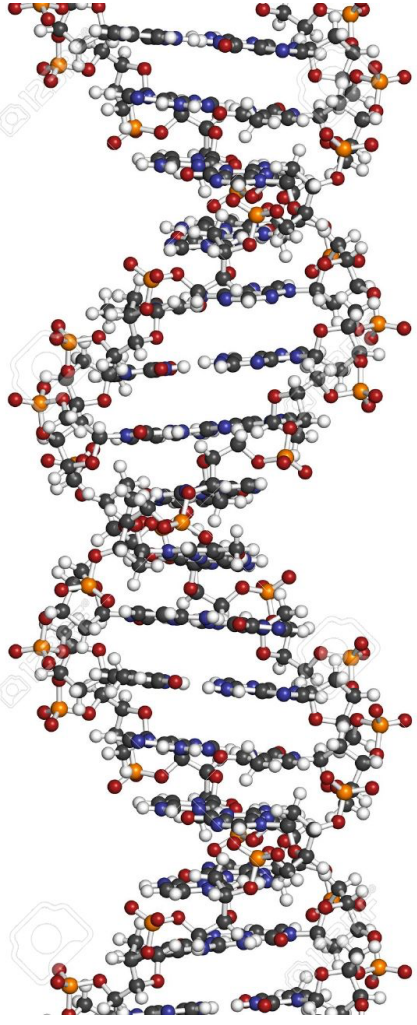
**Figure 2**




Transition from chaos to periodicity in the chaotic dynamics model (CDM) of bacteriocin AS-48 (PDB ID 1e68). (a) The butterfly wing pattern for one of 70 chaotic oscillators. (b) The solution of original 70 chaotic oscillators. (c) The periodic orbit of the ILDM for bacteriocin AS-48. (d) Bacteriocin AS-48 induced Hopf bifurcation from chaos. All of 70 nonlinear oscillators are in one lag synchronized periodic orbit.

# Terminology

- Gene – A variable in the genome
- Genotype – A string of symbols in the genome
- Phenotype – The decoding of a genome
- Locus – A position in the genome ($i^{th}$ position in a string)
- Allele – A value the variable (gene) can take on.
- Epistasis – interdependence of genes (nonlinearity)
- Encoding – A mapping of a set of features into the genome.
- Biological genes can be independent of loci.



# Schemata

Developing a theory of Genetic Algorithms

John Holland, Inventor of the GA.

The following slides are based on material from Goldberg, D. *Genetic Algorithms*, 1989 and Holland J, *Adaptation in Natural and Artificial Systems*, 1993

# GAs: Schemata

Consider a sample problem:

Encoding: integers as bitstrings.
Integers here are the phenotype and bitstrings are the genotype.
We want to maximise the bitstrings according to the fitness function:

$$F(x) = x^2$$

# GAs: Instructive Example

$$F(x) = x^2$$

| *i* | Initial Pop. | Phenotype |
|---|---|---|
| 1 | 01101 | 13 |
| 2 | 11000 | 24 |
| 3 | 01000 | 8 |
| 4 | 10011 | 19 |

# GAs: Instructive Example

$$F(x) = x^2$$

| *i* | Initial Pop. | Phenotype | Fitness (*f*) |
|---|---|---|---|
| 1 | 01101 | 13 | 169 |
| 2 | 11000 | 24 | 576 |
| 3 | 01000 | 8 | 64 |
| 4 | 10011 | 19 | 361 |

# Roulette Selection

$$P(i) = \frac{f_i}{\sum_{i}^{n} f_i}$$

| *i* | Initial Gen. | Phenotype | Fitness (*f*) | *P(i)* |
|---|---|---|---|---|
| 1 | 01101 | 13 | 169 | 0.14 |
| 2 | 11000 | 24 | 576 | 0.49 |
| 3 | 01000 | 8 | 64 | 0.06 |
| 4 | 10011 | 19 | 361 | 0.31 |

# After selection

| *i* | 2nd Gen. | Phenotype | Fitness (*f*) |
|---|---|---|---|
| 1 | 01101 | 13 | 169 |
| 2 | 11000 | 24 | 576 |
| 3 | 11000 | 24 | 576 |
| 4 | 10011 | 19 | 361 |

# After selection, 1-point crossover

| *i* | 2nd Gen. | Phenotype | Fitness (*f*) |
|---|---|---|---|
| 1 | 0110 \| 1 | 13 | 169 |
| 2 | 1100 \| 0 | 24 | 576 |
| 3 | 11 \| 000 | 24 | 576 |
| 4 | 10 \| 011 | 19 | 361 |

After selection, crossover $(P = 1.0)$,

and mutation $\left(P = \frac{1}{1000}\right)$

No mutation in this case.

| *i* | 2nd Gen. | Phenotype | Fitness (*f*) |
|---|---|---|---|
| 1 | 0110 \| 0 | 12 | 144 |
| 2 | 1100 \| 1 | 25 | 625 |
| 3 | 11 \| 011 | 27 | 729 |
| 4 | 10 \| 000 | 16 | 256 |

# GAs: Schemata

What information about the search space does this table contain?

| Bitstring | Fitness |
|---|---|
| 01101 | 169 |
| 11000 | 576 |
| 01000 | 64 |
| 10011 | 361 |

# GAs: Schemata

What information about the search space does this table contain?

Notice that bitstrings starting with 1s have higher fitness.

| Bitstring | Fitness |
|---|---|
| 01101 | 169 |
| 11000 | 576 |
| 01000 | 64 |
| 10011 | 361 |

# GAs: Schemata

1. Similarity among strings in the population.

2. A causal relationship between the strings and the fitness function.

Schemata capture this.

| Bitstring | Fitness |
|---|---|
| 01101 | 169 |
| 11000 | 576 |
| 01000 | 64 |
| 10011 | 361 |

# GAs: Schemata

1. Similarity among strings in the population.

2. A causal relationship between the strings and the fitness function.

Schemata capture this.

| Bitstring | Fitness |
|---|---|
| 01101 | 169 |
| 11000 | 576 |
| 01000 | 64 |
| 10011 | 361 |

Schemata define functional equivalence classes.

# GAs: Schemata

Schemata are defined over the string alphabet plus a metasymbol '*'

'*' is just a wildcard (not a Kleene star if you are in CS500).

# GAs: Schemata

Schemata (or similarity templates)

define equivalence classes.

$$10 * 01$$

# GAs: Schemata

Schemata (or similarity templates) define equivalence classes.

$$10 * 01 = \{10101, 10001\}$$

# GAs: Schemata

Schemata (or similarity templates) define equivalence classes.

$$10 * 01 = \{10101, 10001\}$$

$$*000*$$

# GAs: Schemata

Schemata (or similarity templates) define equivalence classes.

$$10 * 01 = \{10101, 10001\}$$

$$*000* = \{00000, 00001, 10000, 10001\}$$

# GAs: Schemata

Schemata (or similarity templates) define equivalence classes.

$$10 * 01 = \{10101, 10001\}$$

$$*000* = \{00000, 00001, 10000, 10001\}$$

$$0 * 1 * *$$

# GAs: Schemata

Schemata (or similarity templates) define equivalence classes.

$$10 * 01 = \{10101, 10001\}$$

$$*000* = \{00000, 00001, 10000, 10001\}$$

$0 * 1 * *$ All strings of length 5 with a 0 in the first position and a 1 in the third.

# GAs: Schemata

How many schemata are there
for an alphabet with cardinality $k$
and genome length $N$?

# GAs: Schemata

How many schemata are there
for an alphabet with cardinality $k$
and genome length $N$?

$$(k+1)^N$$

# GAs: Schemata

We can bound the number of schemata in a population.

# GAs: Schemata

We can bound the number of schemata in a population.

In our example:

The number of schemata for an individual genome is $2^5$.

because it can take on its actual value or the wildcard.

(2 values)

# GAs: Schemata

We can bound the number of schemata in a population.

In our example:

The number of schemata for an individual genome is $2^5$.
because it can take on its actual value or the wildcard.

If the population has $n$ individuals

there will be at most $n2^2$ schemata.

# GAs: Schemata

So what? How does this help us?

# GAs: Schemata

So what? How does this help us?

Can think in terms of useful classes of genomes, since some of the variation won't matter.

Holland makes an optimality argument for GAs using schemata.

# GAs: Schemata

So what? How does this help us?

Can think in terms of useful classes of genomes, since some of the variation won't matter.

# Schemata

Take a couple of minutes to discuss with your neighbour the effect on

Schemata in the current population of:

- Reproduction
- Crossover
- Mutation

# GAs: Schemata

First we consider reproduction:

# GAs: Schemata

First we consider reproduction:

Schemata with higher fitness tends to increase each generation. No new schemata appear.

# GAs: Schemata: Reproduction

First we consider reproduction:

Schemata with higher fitness tends to increase each generation. No new schemata appear.

Formally, for a schema $S$ in population $A$,

The number of individuals with schema $S$ at time $t+1$, given the number at time $t$ is

$$m_S(t+1) = m_S(t)\frac{f(S)}{\text{mean}(f(A))}$$

# GAs: Schemata: Reproduction

Consider a schema with fitness $c \times \text{mean}(A)$, and $c > 1$

# GAs: Schemata: Reproduction

Consider a schema with fitness $c \times \text{mean}(A)$, and $c > 1$

We can rewrite the increase in representation for this schema, $S$, as:

$$m_S(t+1) = m_S(t)\left[\frac{c \times \text{mean}(A)}{\text{mean}(A)}\right]$$

# GAs: Schemata: Reproduction

We can rewrite the increase in representation for this schema, $S$, as:

$$m_S(t+1) = m_S(t)\left[\frac{c \times \text{mean}(A)}{\text{mean}(A)}\right]$$

$$m_S(t+1) = c \times m_S(t)$$

# GAs: Schemata: Reproduction

We can rewrite the increase in representation for this schema, $S$, as:

$$m_S(t+1) = m_S(t)\left[\frac{c \times \text{mean}(A)}{\text{mean}(A)}\right]$$

$$m_S(t+1) = c \times m_S(t)$$

$$m_S(t) = c \times c \times \cdots \times c \times m_S(0)$$

# GAs: Schemata: Reproduction

We can rewrite the increase in representation for this schema, $S$, as:

$$m_S(t+1) = m_S(t)\left[\frac{c \times \text{mean}(A)}{\text{mean}(A)}\right]$$

$$m_S(t+1) = c \times m_S(t)$$

$$m_S(t) = c \times c \times \cdots \times c \times m_S(0) = c^t$$

**Exponential**

# GAs: Schemata

Now consider the effect of 1-point crossover on schemata.

# GAs: Schemata

Now consider the effect of 1-point crossover on schemata.

Schemata survive if they are not cut by the crossover

For example:

$1***0$ is ~~less~~ more likely than $***10*$ to be destroyed.

# Formally:

Probability of disruption: $P_D = P_\times \dfrac{D(S)}{L_S - 1}$ where,

1 fewer crossover sites than genes

$P_\times$ is the crossover probabilty,

$D(S)$ is the defining length of schema $S$,

$L$ is the string/genome length.

Defining length: distance between the first and last non-wildcard symbols.

GAs: Schemata

# How about mutation?

# GAs: Schemata

Mutation is more likely to destroy "higher order" schemata.

The order is the number of fixed (non-wildcard) symbols.

# GAs: Schemata

Mutation is more likely to destroy "higher order" schemata.
The order is the number of fixed (non-wildcard) symbols.

More formally,

Probability of disruption: $P_D = \left(1 - (1 - P_{\text{mut}})^{O(S)}\right)$ where,

$P_{\text{mut}}$ is the mutation probability,
$O(S)$ is the order of the schema $S$.

# GAs: Schemata

**Also called the fundamental Theorem of Genetic Algorithms**

The schema theorem:

Fit schema with lower defining length and lower order increase exponentially in the population over time.

These are called "building blocks"

# GAs: Schemata

**Also called the fundamental Theorem of Genetic Algorithms**

Schema Theorem

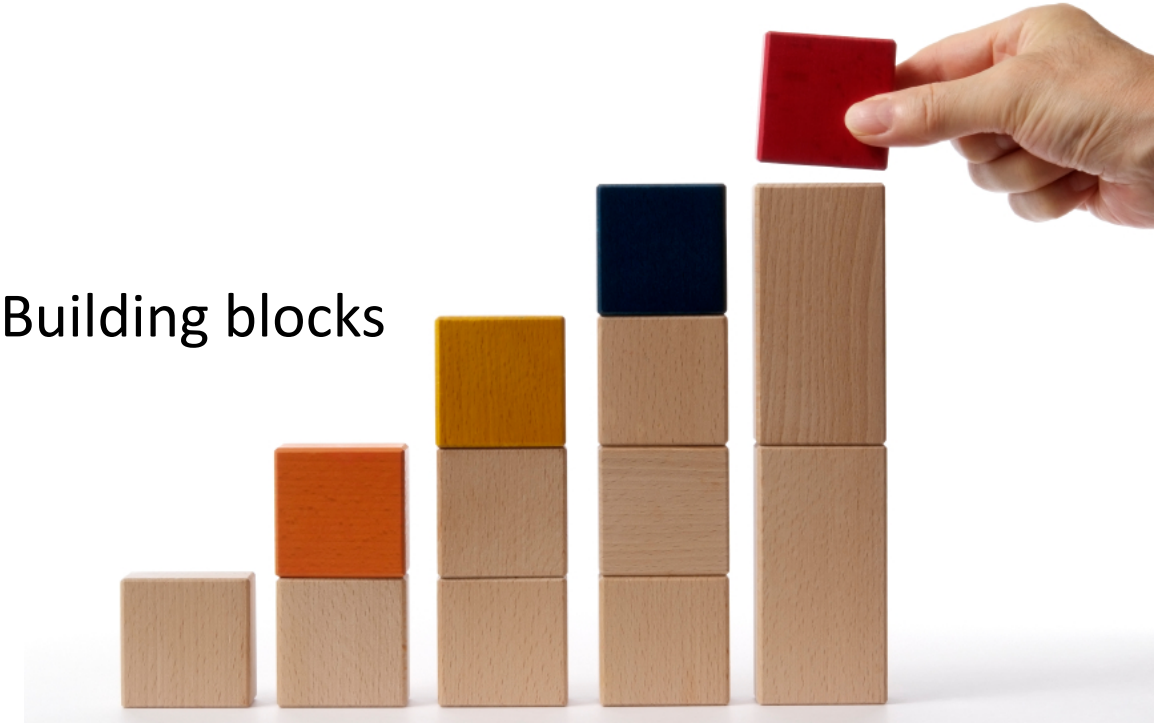
More formally,

$$m_S(t+1) \geq m_S(t) \frac{f(S)}{\text{mean}(S)} \left[ 1 - P_\times \frac{D(S)}{L-1} - O(S) P_{\text{mut}} \right]$$

# GAs: Schemata

Building block hypothesis: A genetic algorithm seeks optimal performance through the juxtaposition of short, low-order, high- performance schemata, called the building blocks.

Premature convergence is the main challenge. Building blocks can cause premature convergence.

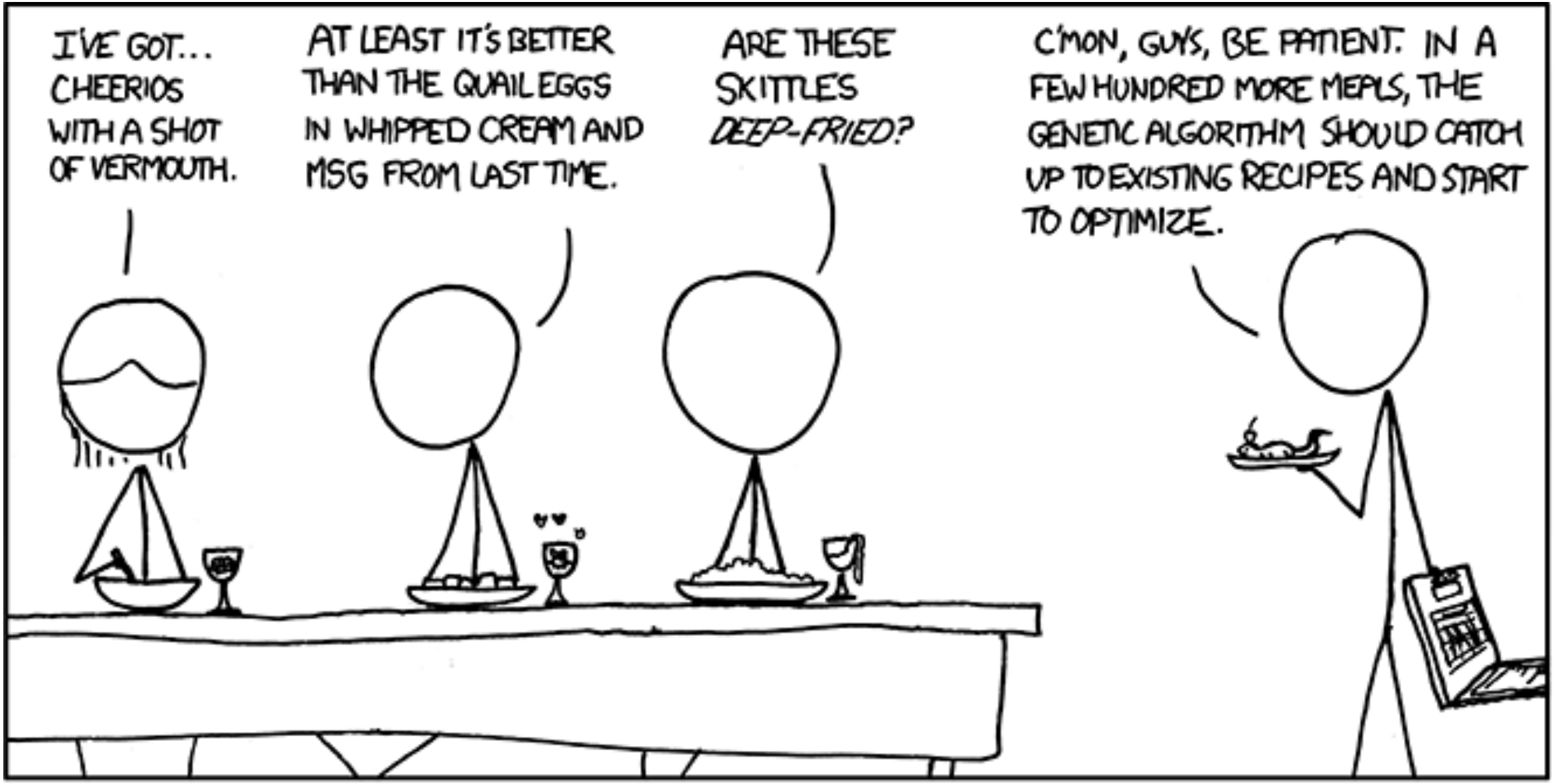


# A Royal Road Problem: Testing the Building Block Hypothesis

$$
\begin{aligned}
s_1 &= 11111111**********; & c_1 &= 8 \\
s_2 &= ********11111111********; & c_2 &= 8 \\
s_3 &= ****************11111111********; & c_3 &= 8 \\
s_4 &= ************************11111111********; & c_4 &= 8 \\
s_5 &= ********************************11111111********; & c_5 &= 8 \\
s_6 &= ****************************************11111111********; & c_6 &= 8 \\
s_7 &= ************************************************11111111********; & c_7 &= 8 \\
s_8 &= ****************************************************************11111111; & c_8 &= 8 \\
s_9 &= 1111111111111111**********; & c_9 &= 16 \\
s_{10} &= ****************1111111111111111**********; & c_{10} &= 16 \\
s_{11} &= ********************************1111111111111111**********; & c_{11} &= 16 \\
s_{12} &= ****************************************************1111111111111111; & c_{12} &= 16 \\
s_{13} &= 11111111111111111111111111111111**********; & c_{13} &= 32 \\
s_{14} &= ********************************11111111111111111111111111111111; & c_{14} &= 32 \\
s_{15} &= 1111111111111111111111111111111111111111111111111111111111111111; & c_{15} &= 64
\end{aligned}
$$

Figure 1: Example Royal Road Function. $F(x) = \sum_{s \in S} c_s \sigma_s(x)$, where $x$ is a bit string, $c_s$ is a value assigned to the schema $s$ (here, $c_s = order(s)$), and $\sigma_s(x) = \begin{cases} 1 & \text{if } x \text{ is an instance of } s \\ 0 & \text{otherwise.} \end{cases}$

Mitchell, Melanie, Stephanie Forrest, and John H. Holland. "The royal road for genetic algorithms: Fitness landscapes and GA performance." *Proceedings of the first european conference on artificial life*. 1992.




WE'VE DECIDED TO DROP THE CS DEPARTMENT FROM OUR WEEKLY DINNER PARTY HOSTING ROTATION.

https://xkcd.com

# Logistics

- Midterm exam - March 10th.
- Midterm review moved to March 6th.

- Project 2 is due on March 10th.
- Project reviews are due on March 20th.
- The in class competition will be on March 20th.

- Transitioning from Genetic Algorithms to Cellular Automata
- We will be using genetic algorithms in the next project to explore cellular automata and game theory.

# Theory of Self-Reproducing Automata - John von Neumann

- David Shubsda
- Joshua Ridens

- Turn in your review forms in to Bianca at the end of the presentation.
- I will put David and Joshua’s slides on the course website.

# GAs: k-armed bandits


Suppose you can play $k$ slot machines.

The $i^{\text{th}}$ machine pays jackpot following a Gaussian probability distribution with

mean $= \mu_i$, and variance $= \sigma_i^2$

# GAs: k-armed bandits


Suppose you can play $k$ slot machines.

The $i^{\text{th}}$ machine pays jackpot following a Gaussian probability distribution with

mean $= \mu_i$, and variance $= \sigma_i^2$

The challenge is to win the most money possible (or lose the least) over time.

# GAs: k-armed bandits

Reason about the best trade-off

# GAs: k-armed bandits

Reason about the best trade-off


## A strategy:

Given a maximum of N pulls at the arm we could try:

Allocating an equal number, $n$, of pulls, where $(kn < N)$, to each arm, then use all the remaining time to pull on the arm with the most payouts.

# GAs: k-armed bandits

Reason about the best trade-off


Given $N, \mu_i,$ and $\sigma_i^2,$

we can define a loss function, $\mathrm{L}(N, n)$.

# GAs: Optimality of Trial Allocation

- There is a trade-off between sampling near the best observed and exploring the fitness landscape.
- Loss due to searching near the currently known optimum is due to *sampling error*.
- Loss due to choosing areas that are not known to be good we might call *performance loss*.

# GAs: Optimality of Trial Allocation

- The presence of a trade-off suggests an optimisation problem.

# GAs: k-armed bandits


Reason about the best trade-off

Given $N, \mu_i,$ and $\sigma_i^2,$

we can define a loss function, $\mathrm{L}(N, n)$.

$$\mathrm{L}(N, n) = |\mu_1 - \mu_2| \left[ (N - n) q(n) + n(1 - q(n)) \right]$$

The 2-arm case generalises

De Jong, K. A. *An analysis of the behaviour of a class of genetic algorithms*. Diss. PhD thesis, University of Michigan, 1975.

# GAs: k-armed bandits

$$\mathrm{L}(N, n) = |\mu_1 - \mu_2| \left[ (N - n) q(n) + n(1 - q(n)) \right]$$

Where $q(n)$ is the probability that the worst arm is the best arm after $n$ pulls.

In other words $q(n)$is the probability that you got unlucky as a function of the explore/exploit trade-off.

De Jong, K. A. *An analysis of the behaviour of a class of genetic algorithms*. Diss. PhD thesis, University of Michigan, 1975.

$q(n)$ decreases exponentially with $n$. To minimise $q(n)$ we allocate $O(e^n)$ trials to the known best.

"The ratio of trials of the observed best to the second best grows exponentially."

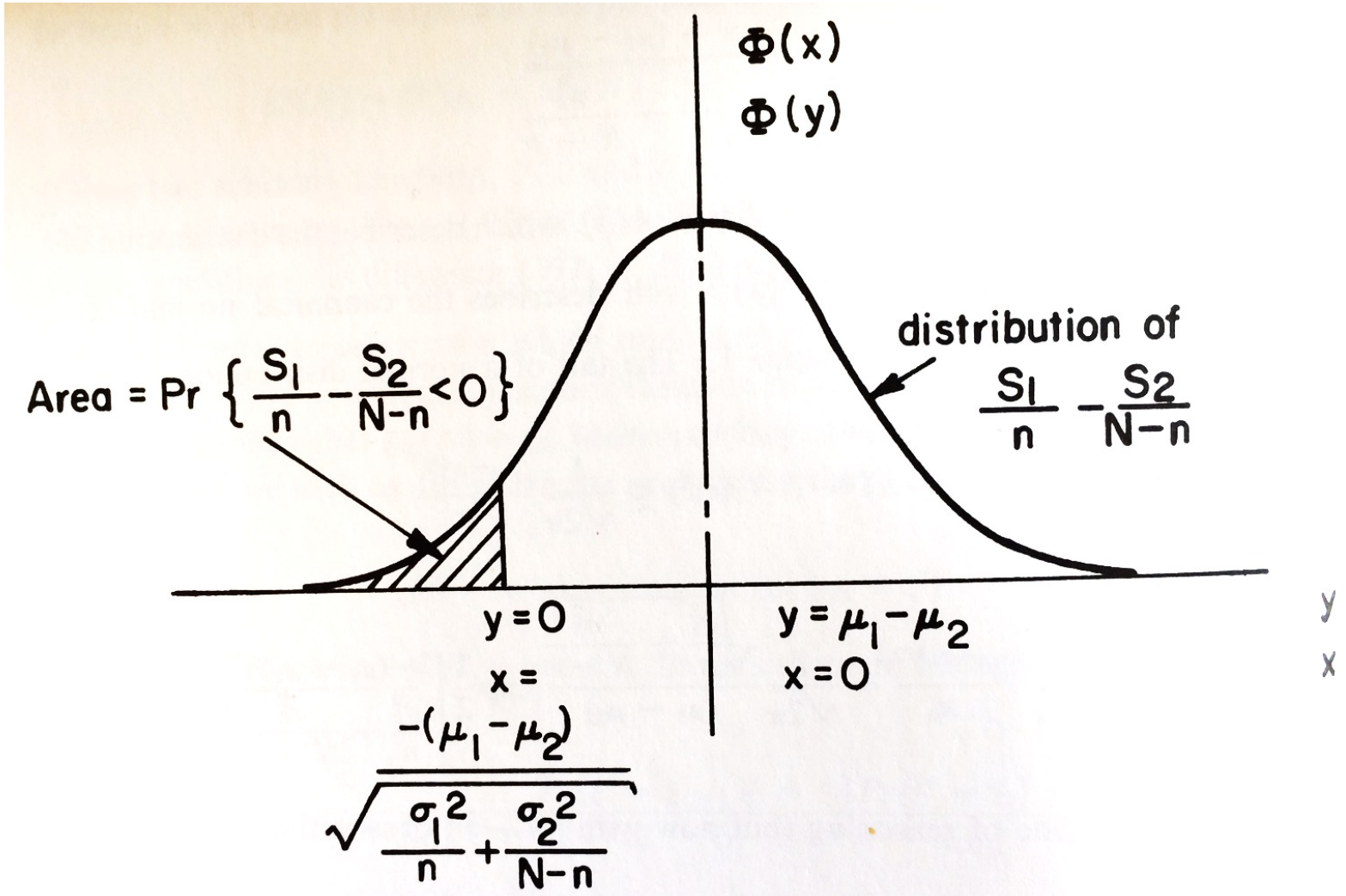



*Fig. 11. The convolution of* $\frac{S_1}{n}$ *with* $\frac{S_2}{N-n}$

Holland J, *Adaptation in Natural and Artificial Systems*, 1993

# GAs: k-armed bandits

Holland showed that allocating an exponential number of pulls to the best performing slot machine minimises the loss function $L(N, n)$.

This exponential allocation to exploitation best solves the problem.

Holland, J., "Genetic algorithms and the optimal allocation of trials." *SIAM Journal on Computing* 2.2 (1973): 88-105.

# GAs: k-armed bandits

Holland showed that allocating an exponential number of pulls to the best performing slot machine minimises the loss function $L(N, n)$.

This exponential allocation to exploitation best solves the problem.

**What does this have to Do with the Fundamental Theorem?**

# GAs: k-armed bandits

Holland showed that allocating an exponential number of pulls to the best performing slot machine minimises the loss function $L(N, n)$.

This exponential allocation to exploitation best solves the problem.

**What does this have to Do with the Fundamental Theorem?**

**The fundamental theorem shows that GAs allocate exponentially increasing samples to known best solutions.**

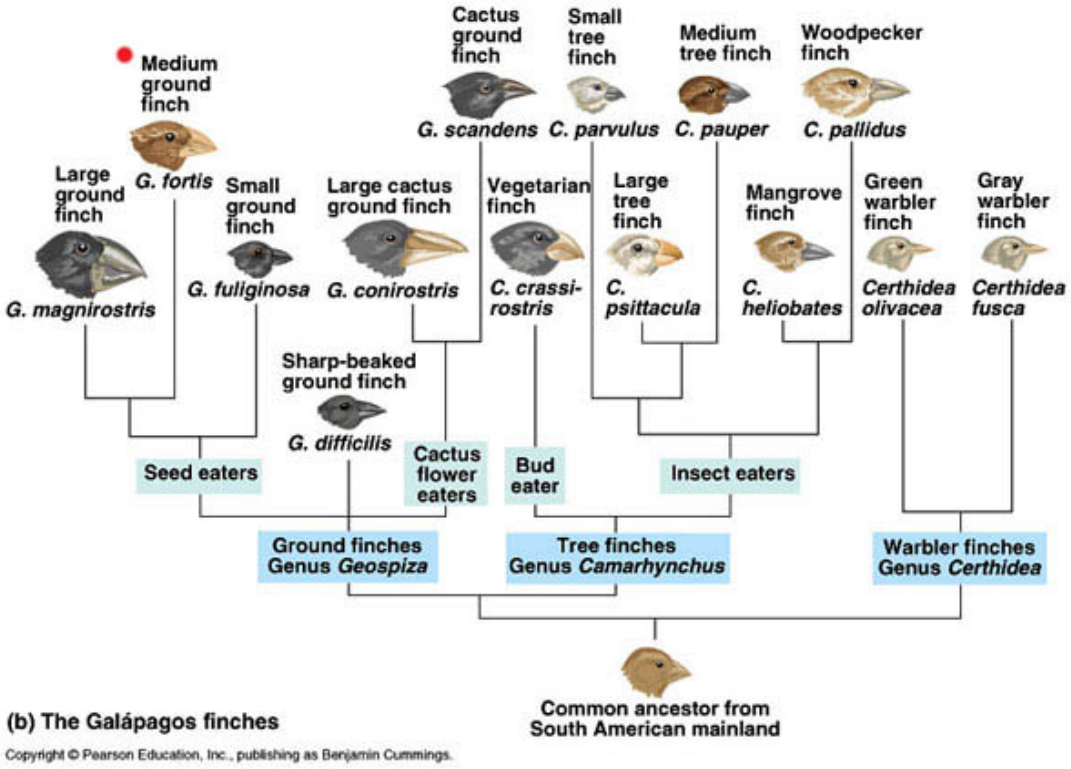
# Diversity

- A pitfall of GAs is that they may converge too early on a local maxima.
- If the population were infinite this wouldn't be a problem.
- The smaller the population the more likely they are to converge prematurely.
- Building blocks can increase premature convergence for some problems.

# Schemata: Implicit Parallelism

Holland also showed in 1985 that if $K$
is the number of stringsprocessed each generation
and $N$ is the number of schemata
Then $N \in O(K^3)$.

# Diversity: Island GAs




(b) The Galápagos finches



Recall Darwin's famous finches. The diversity is in part due to evolution on islands.

# Diversity: Island GAs

What would this look like as an algorithm?

# Diversity: Elitism

- Diversity is great for avoiding local minima.
- How do we keep from moving too far from regions we know are good.
- Roulette selection can easily discard the most fit individuals.

# Diversity: Elitism

- Diversity is great for avoiding local minima.
- How do we keep the population from forgetting about regions we know are good.
- Roulette selection can easily discard the most fit individuals.
- High mutation rates can move the whole population to a lower local maxima.
- Copying the highest fitness individual into the next generation unchanged is called *elitism*.

# Diversity: Elitism

- Elitism guarantees that the GA will converge*

(Applies to simulated annealing and artificial immune system optimization as well)

*Villalobos-Arias et al, “Asymptotic Convergence of Some Metaheuristics Used for Multiobjective Optimization”, *Foundations of Genetic Algorithms*, 2005

# Selection Pressure: Tournaments

- Selection pressure is a measure of how harsh we make the world.
- At one extreme only the very fittest individual would survive.
- At the other extreme everyone survives.

# Selection Pressure: Tournaments

- If selection pressure is too high the GA will converge prematurely on a local minima.

- If selection pressure is too low the GA will not converge at all.

# Selection Pressure: Tournaments

- Tournament selection is one way to tune the selection pressure.

# Evolving Programs

Project 2

# Project 2: Core Wars

- Alexander Dewdney, Mathematician and Computer Science
- Wrote a series of articles called Mathematical Recreation for Scientific American
- One of these was Core Wars
- Inspired by a real life incident (Creeper and Reaper).
- Christopher Langton invited Dewdney to present core wars
  At the first Alife conference.
- Playing with Core Wars was once considered dangerous.
- Capable of self-mutation…

Battle Info

Core Size: 8000 Max Cycles: 80000

Battle Status: Stopped

Completion: 0% 0

Tournament Type: Round Robin Show Results

Round 1 of 1 Matchup 1 of 6

Max Processes: 8000

| | | |
|---|---|---|
| Imp | 0% | 1 |
| Mice | 0% | 1 |
| Midget | | |
| Piper | | |

# Some x86 Assembly language

| Address | Instruction | Register or RAM Address |
|---|---|---|
| 77E814EE | mov | esi,dword ptr [edi+8] |
| 77E814F1 | mov | dword ptr [ebp+64h], 0Ah |
| 77E814F8 | add | esi, 4Ah |
| 77E814FB | jmp | 77E7E91A |
| 77E81500 | push | ebx |
| 77E81501 | xor | ebx,ebx |
| 77E81503 | cmp | ecx,ebx |
| 77E81505 | push | esi |
| 77E81506 | push | edi |

```
;redcode
;author: T77
;name: Example1
;assert CORESIZE=8000 && MAXLENGTH > 100
MOV 0, 1
```

# # - immediate addressing

```
;redcode
;author: T77
;name: Example2
;assert CORESIZE=8000 && MAXLENGTH > 100
ADD #4, 3
MOV 2, @2
JMP -2
DAT #0, #0
```

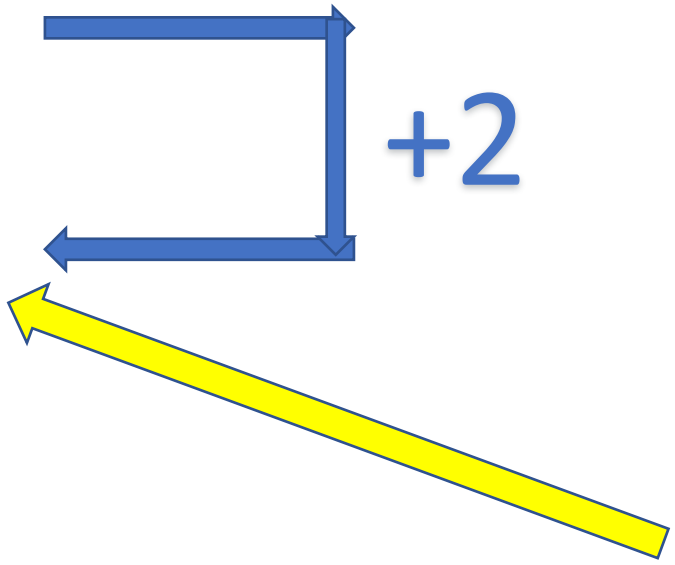
Program counter is here

```
;redcode
;author: T77
;name: Example2
;assert CORESIZE=8000 && MAXLENGTH > 100
ADD #4, 3
MOV 2, @2
JMP -2
DAT #0, #4
```

Program counter is here

# @ - indirect addressing

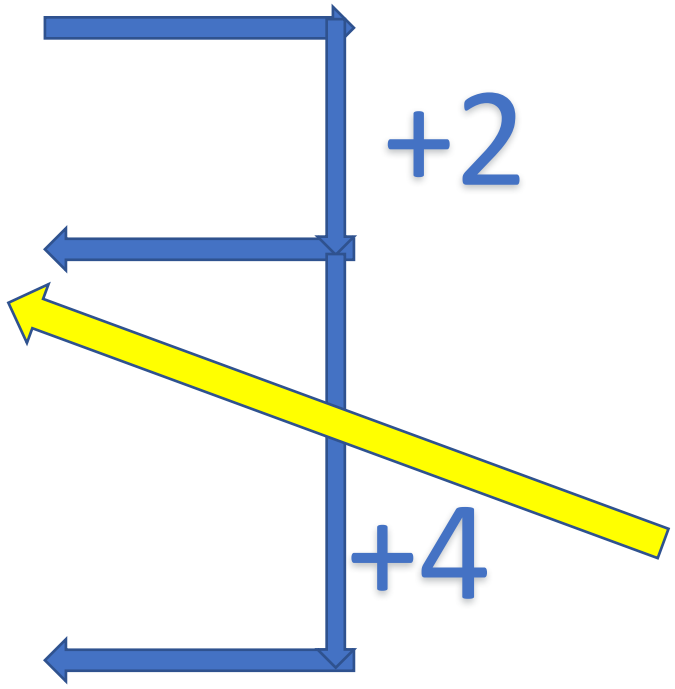
```
;redcode
;author: T77
;name: Example2
;assert CORESIZE=8000 && MAXLENGTH > 100
ADD #4, 3
MOV 2, @2
JMP -2
DAT #0, #4
.
.
.
DAT #0, #4
```




B field points here

# @ is indirect addressing

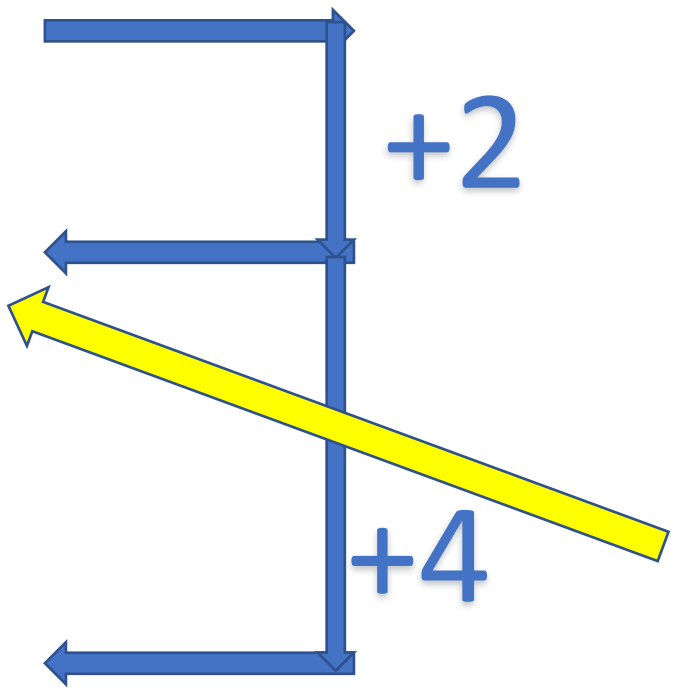
```
;redcode
;author: T77
;name: Example2
;assert CORESIZE=8000 && MAXLENGTH > 100
ADD #4, 3
MOV 2, @2
JMP -2
DAT #0, #4
.
.
.
DAT #0, #4
```




B field points here

# This program places dats separated by 4 addresses forever.

```
ADD #4, 3
MOV 2, @2
JMP -2
DAT #0, #4
.
.
.
DAT #0, #4
```




B field points here

```
;redcode
;author: T77
;name: Example3
;assert CORESIZE=8000 && MAXLENGTH > 100
spl  0
mov  2,  <-1
jmp  -1,  -1
```

Create new thread
And continue with next instruction

```
;redcode
;author: T77
;name: Example3
;assert CORESIZE=8000 && MAXLENGTH > 100
spl  0
mov  2,  <-1
jmp  -1,  -1
```

< Indirect with predecrement