Using an Artificial Regulatory Network to Investigate Neural Computation

W. Garrett Mitchener

College of Charleston

January 16, 2015
Section 1

Evolution and Computing
Computation: sequential register machine

fastrand:

move $2, $4 \quad / R2 = S \ (\text{arg passed in R4})
li $8, 33614 \quad / R8 = 2 \times \text{constant A}
multu $2, $8 \quad / HI, LO = A \times S
mflo $9 \quad / R9 = Q = \text{bits 00..31 of } A \times S
srl $9, $9, 1 \quad / R10 = P = \text{bits 32..63 of } A \times S
mfhi $10 \quad / R2 = S' = P + Q
addu $2, $9, $10 \quad / \text{handle overflow (rare)}
bltz $2, overflow \quad / \text{return (result in R2)}
j $31

overflow:
sll $2, $2, 1 \quad / \text{zero bit 31 of } S'
srl $2, $2, 1
addiu $2, 1 \quad / \text{increment } S'
j $31 \quad / \text{return (result in R2)
Computation: Expression trees

- Expression tree
- Evaluated by reductions

\[ x = 5 \times 3^2 + 4 \]
Computation: Minimalist syntax

John gave Fred a book
Computation: Minimalist syntax

Who gave Fred a book?
Bio-computing: Genes and transcription regulation

- Molecules in solution: ions, proteins, DNA, RNA, …
- Many simultaneous reactions

lac operon

- lacI repressor
- lacZ enzyme
- lacY
- lacA

lacI disabled by lactose

Molecules in solution: ions, proteins, DNA, RNA, …

Many simultaneous reactions
Biological Computation: Action potentials

- Electrical pulses sent via ions, neurotransmitters
- Detect coincidences, crossing of threshold
- Vast neural networks, active cardiac tissue
Section 2

Utrecht Machine
Utrecht Machine: State

- **patterns**: transcription factors (integer)
- **activation**: molecule count (integer)

<table>
<thead>
<tr>
<th>$p$</th>
<th>$A[p]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Utrecht Machine: Instructions

\[ j : A[p_{\text{switch}}] \geq \theta \Rightarrow \uparrow A[p_{\text{up}}] \downarrow A[p_{\text{down}}] \]

- \( j \): instruction number
- \( p_{\text{switch}} \): switch pattern
- \( \theta \): threshold
- \( p_{\text{up}} \): pattern to increment
- \( p_{\text{down}} \): pattern to decrement

If the activation of \( p_{\text{switch}} \) is at least \( \theta \), add one to the activation of \( p_{\text{up}} \) and subtract one from the activation of \( p_{\text{down}} \).
UM Example: NAND

0: $A[0] \geq 0 \Rightarrow \uparrow A[0] \downarrow A[7]$


Output is $A[6] \geq 3$

Stop when $A[0] \geq 4$
UM Example: NAND

0: $A[0] \geq 0 \Rightarrow \uparrow A[0] \downarrow A[7]$

Stop when $A[0] \geq 4$
UM Example: NAND

0: \(A[0] \geq 0 \Rightarrow \uparrow A[0] \downarrow A[7]\)

Input via \(A[4]\) and \(A[5]\)
Output is \(A[6] \geq? 3\)
Stop when \(A[0] \geq 4\)
Show only efficacious genes and arrows
Biological genetic code

<table>
<thead>
<tr>
<th>1st base</th>
<th>U</th>
<th>2nd base</th>
<th>A</th>
<th>G</th>
<th>3rd base</th>
</tr>
</thead>
<tbody>
<tr>
<td>UUU</td>
<td>UCU</td>
<td>UAU</td>
<td>UGU</td>
<td>C</td>
<td>U</td>
</tr>
<tr>
<td>UUC</td>
<td>UCC</td>
<td>UAC</td>
<td>CGC</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>UUA</td>
<td>UCA</td>
<td>UAA</td>
<td>UGA</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>UUG</td>
<td>UCG</td>
<td>UAG</td>
<td>UGG</td>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>CUU</td>
<td>CUC</td>
<td>CAU</td>
<td>CGU</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>CUC</td>
<td>CCC</td>
<td>CAC</td>
<td>CGC</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>CUA</td>
<td>CCA</td>
<td>CAA</td>
<td>CGA</td>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>CUG</td>
<td>CCG</td>
<td>CAG</td>
<td>CGG</td>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>AUU</td>
<td>ACU</td>
<td>AAU</td>
<td>AGU</td>
<td>U</td>
<td>U</td>
</tr>
<tr>
<td>AUC</td>
<td>ACC</td>
<td>AAC</td>
<td>AGC</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>AUA</td>
<td>ACA</td>
<td>AAA</td>
<td>AGA</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>AUG</td>
<td>ACG</td>
<td>AAG</td>
<td>AGG</td>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>GUU</td>
<td>GCU</td>
<td>GAU</td>
<td>GGU</td>
<td>U</td>
<td>U</td>
</tr>
<tr>
<td>GUC</td>
<td>GCC</td>
<td>GAC</td>
<td>GGC</td>
<td>G</td>
<td>G</td>
</tr>
<tr>
<td>GUA</td>
<td>GCA</td>
<td>GAA</td>
<td>GGA</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>GUG</td>
<td>GCG</td>
<td>GAG</td>
<td>GGG</td>
<td>G</td>
<td>G</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1st base</th>
<th>2nd base</th>
<th>3rd base</th>
</tr>
</thead>
<tbody>
<tr>
<td>UUU</td>
<td>Phenylalanine</td>
<td>UCU</td>
</tr>
<tr>
<td>UUC</td>
<td>Leucine</td>
<td>UCC</td>
</tr>
<tr>
<td>UUA</td>
<td>Serine</td>
<td>UCA</td>
</tr>
<tr>
<td>UUG</td>
<td>Cysteine</td>
<td>UCG</td>
</tr>
<tr>
<td>CUU</td>
<td>Proline</td>
<td>CUC</td>
</tr>
<tr>
<td>CUC</td>
<td>Histidine</td>
<td>CCC</td>
</tr>
<tr>
<td>CUA</td>
<td>Glutamine</td>
<td>CCA</td>
</tr>
<tr>
<td>CUG</td>
<td>Glutamic acid</td>
<td>CCG</td>
</tr>
<tr>
<td>AUU</td>
<td>Isoleucine</td>
<td>ACU</td>
</tr>
<tr>
<td>AUC</td>
<td>Asparagine</td>
<td>ACC</td>
</tr>
<tr>
<td>AUA</td>
<td>Threonine</td>
<td>ACA</td>
</tr>
<tr>
<td>AUG</td>
<td>Methionine</td>
<td>ACG</td>
</tr>
<tr>
<td>GUU</td>
<td>Valine</td>
<td>GCU</td>
</tr>
<tr>
<td>GUC</td>
<td>Alanine</td>
<td>GCC</td>
</tr>
<tr>
<td>GUA</td>
<td>Aspartic acid</td>
<td>GCA</td>
</tr>
<tr>
<td>GUG</td>
<td>Glutamic acid</td>
<td>GCG</td>
</tr>
</tbody>
</table>

W. Garrett Mitchener (C of C)
Utrecht Machine: Genetic code, majority-of-three

Two robust codons:
- 000 decodes to 0
- 111 decodes to 1
- Takes two substitutions to change interpretation

Six fragile codons:
- 001, 010, 100: Decode to 0
- 110, 101, 011: Decode to 1
- One substitution can change
Utrecht Machine: Genetic code

Genome & instructions

0: $A[0] \geq 0 \Rightarrow \uparrow A[0] \downarrow A[7]$
UM: Point substitutions
UM: Deletions
UM: Duplications
UM: Crossover
Communication problem setup

Input

Sender

Receiver

Output

1

9

3

10

4

13

10

1

12

10

0

2

0

10

W. Garrett Mitchener (C of C)
Example solution: Sample Run 47, agent 377513
Selective breeding

- Breed 200 children from highest 100
- Breed 100 children from whole population
- New population = 200 old agents with the highest rating + 300 new agents
Evolutionary overview

Maximum score of first $n$ agents created as a function of $n$

V1 8403
Running to saturation

One perfect solution takes many generations to saturate the population.
Implicit selection for robustness

1. First perfect solution found
2. Runs to saturation
3. Competition for 200 survivor slots
4. Favors agents whose offspring are more likely to have perfect scores
5. $\iff$ favors robustness against mutation and crossover
Measuring robustness: Deleterious mutations

- Mutations that lower performance are *deleterious*
- Measure of robustness
  - $g =$ base genome with perfect score
  - $M(g) =$ all possible mutations
  - $G =$ set of genomes one mutation away from $g$
    \[ G = \{ g \oplus m \mid m \in M(g) \} \]
  - How many genomes in $G$ score less than $g$?
  - What fraction of genomes in $G$ score less than $g$?
Sample Run 143: Deleterious mutations
Section 3

Sample Run 7: Synaptic Code
Sample Run 7: Synaptic Code

Sample Run 7: Synaptic Codes

Activation times:
\[ t_{11}, t_{01}, t_{10}, t_{00} \]

Synaptic timing gap:
\[ G = \left| t_{\text{uniEBF5}} - t_{\text{uniEBF6}} \right| \]
Sample Run 7: Synaptic Codes

Activation times:

\((t_{11}, t_{01}, t_{10}, t_{00})\)
Sample Run 7: Synaptic Codes

Activation times:

\[ (t_{11}, t_{01}, t_{10}, t_{00}) \]

Synaptic timing gap:

\[ \text{STG} = \min_{u,v} |t_u - t_v| \]
Sample Run 7: Synaptic timing gap distribution

Fraction of perfectly scoring agents with STG of 1, 2, 3 as a function of generation number
Section 4

Sample Run 47: Simplification, Redundancy, Codon bias
First perfect solution.
Sample Run 47, agent 377513

Dangerous extra link is gone
Sample Run 47, agent 377871

Redundancy appears
Codon bias

- two robust codons
- six fragile codons
- Each gene has 22 codons
  - If uniformly random expect 5.5 robust codons per gene
  - If robustness favored expect more
- Consider genes with 6 or more robust codons
Section 5

Sample Run 509: Genome Rearrangement
Sample Run 509: First perfect solution, generation 1361

= inefficacious.  = part of sender.  = part of receiver.  = part of both.
Sample Run 509: Genome Rearrangement

Sample Run 509: Perfect solution from generation 1700

= inefficacious. = part of sender. = part of receiver. = part of both.
Sample Run 509: Perfect solution from generation 2700

= inefficacious. = part of sender. = part of receiver. = part of both.
Sample Run 509: Genome rearrangement

○ = inefficacious. ▢ = part of sender. □ = part of receiver. △ = part of both.
Sample Run 509: Genome rearrangement

= inefficacious. = part of sender. = part of receiver. = part of both.
Sample Run 509: Genome rearrangement

= inefficacious. = part of sender. = part of receiver. = part of both.
About me

E-mail: mitchenerg@cofc.edu
Web site: http://mitchenerg.people.cofc.edu