Evolving Self-Reproducing Functional Programs

Bob McKay & Daryl Essam
School of Computer Science
University of New South Wales /
Australian Defence Force Academy
Outline

• Background
  – Abiogenesis
  – Koza and Pargellis

• Aims

• Language for self-evolution

• Experimental Design

• Experimental Results

• Discussion

• Conclusions

• Further Work
Abiogenesis

• Early work on the chemical process
  – Miller & Urey (1953)

• General agreement on at least the possible outlines of the chemistry

• Presupposes parallel development of increasingly effective search methods, culminating in evolutionary search

• Earlier search methods presuppose resource limitation, although evolutionary search does not
Previous Work in Generating Self-Reproducing Programs

• Von Neumann (1949) derived self-reproducing Turing Machine to draw parallel between life and computational processes

• Koza (1991, 1994) - simulates process of abiogenesis including such issues as resource constraints, etc.

• Programming structures relatively low-level, reflecting the desire to simulate aspects of the physical world

• Pargellis (1996 a,b; 2001) similar
Aim

• Separate out the issues of growth of complexity and emergence of self-reproduction

• Investigate emergence of self-reproduction in languages in which it is simply represented

• (long term) investigate the effects of different search algorithms on the emergence of self-reproduction
Learning Language

- $S \rightarrow M$
- $M \rightarrow \text{IF } M M M$
- $M \rightarrow \text{JOIN } M M$
- $M \rightarrow \text{QUOTE } M$
- $M \rightarrow \text{FUN } M$
- $M \rightarrow X$
- $M \rightarrow \text{UNDEF}$

- If A then B else C
- Catenate funcs
- Return tree M
- Execute M
- Return value of X
- Return undefined
Hand-coded Self-reproducing Program

- (IF X
  - (FUN (QUOTE (JOIN (JOIN (JOIN (QUOTE (IF UNDEF UNDEF UNDEF )) )
    (QUOTE X ))
    (JOIN (QUOTE (FUN (QUOTE (QUOTE UNDEF ))) X ) )
    X ) )
)
)

• )
• )

• )
• )

• )
• )
Experimental Design

- Grammar Guided Genetic Programming
- Series of fitness functions incrementally designed to find solution
- Each fitness function executed for ten runs
- GP parameters chosen to permit sufficiently complex programs, large population to preserve diversity
# GP Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>SPECIFICATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Runs</td>
<td>10</td>
</tr>
<tr>
<td>Max Generations</td>
<td>100</td>
</tr>
<tr>
<td>Population Size</td>
<td>1000/500</td>
</tr>
<tr>
<td>Max depth (initial pop)</td>
<td>12</td>
</tr>
<tr>
<td>Max depth (subsequent)</td>
<td>12</td>
</tr>
<tr>
<td>Tournament size</td>
<td>3</td>
</tr>
<tr>
<td>Crossover Probability</td>
<td>0.9</td>
</tr>
<tr>
<td>Mutation Probability</td>
<td>0.1</td>
</tr>
</tbody>
</table>
Fitness Functions

• 1. difference in size between two trees
• 2. difference in size between two trees, penalty for very small trees
• 3. difference between two trees
• 4. difference between two trees divided by sum of sizes of trees
• 5. proportionate difference between two trees, but artificially set to an arbitrarily large number if the trees did not execute a 'fun' term
• 6. proportionate difference between two trees, but artificially set to an arbitrarily large number if the trees did not execute both a 'fun' term and an 'if' term
Results

• 1. Found un-interesting ‘solutions’: ‘x’, ‘undef’, ‘null’

• 2. Produced large trees similar in size, but not content, to their product

• 3. Quoted functions:
  – (quote (if x x (quote (join x x))))

• 4. Quoted functions of maximum depth:
  – (quote (quote (quote (quote ... (quote x)))))

• 5. Functions in which the ‘fun’ term was applied to null (yielding null)
  – (....(fun null)......)
Results

– 6. Found a self-reproducing program, simpler than the hand-coded example:
  
  • (join (quote
  • (join (quote undef)
  • (fun (if (if x x x) (quote x) (if undef x undef)))
  • (fun (if (if x x x) (quote x) (if undef x undef) ) ) )


Not minimal solution

– Can be simplified by replacing
  – (if x x x) by x
  – (if undef x undef) by undef

• (join (quote
• (join (quote undef)
• (fun (if x (quote x) undef))
• (fun (if x (quote x) undef ) ) )

– Nevertheless far simpler than both the hand-coded solution, and corresponding programs found by Koza and by Pargellis
Parsimony

– Self-reproducing program found in generation 18
– Did not change in subsequent 82 generations of run
– Contrary to common experience with GP: genome tends to grow, particularly after convergence
– Presumably due to indirect parsimony pressure: increased complexity has to be reproduced in both source and destination program
Conclusions

– Within a carefully tailored simple functional system, self-reproduction may be generated by evolutionary means with relatively modest computational resources

– Provides an environment within which the abiogenetic emergence of self-reproduction may be comparatively investigated
Further Work

– Investigate abiogenetic emergence within the same computational framework

– Compare incremental development of evolutionary search with direct evolutionary search

– Aim to cast light on the gradual emergence of evolutionary search without the complications of the emergence of complexity