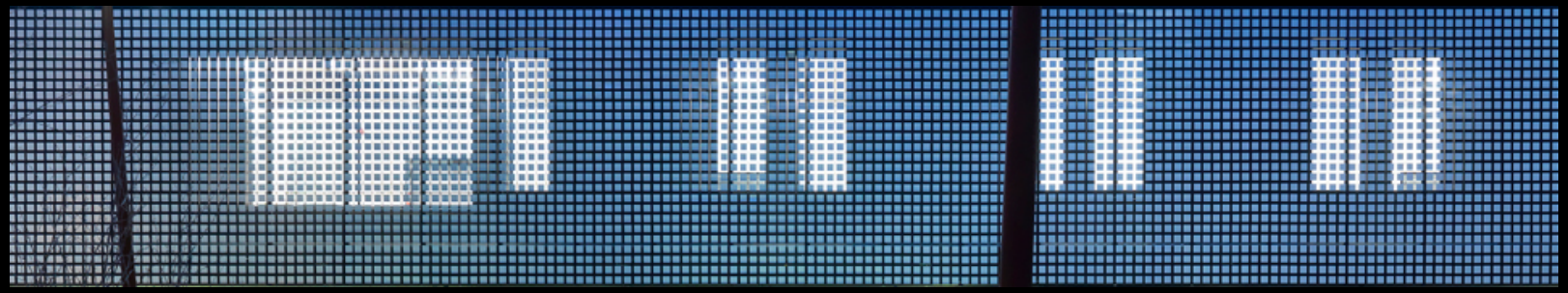


gp with meaning and confidence

colin johnson / university of kent

Schtick

- That techniques from the more formal end of computer science/software engineering have a lot to contribute to genetic programming and related areas.

Overview

- the use of formal measures of program correctness in fitness functions
- the use of scaffolding assumptions to evolve recursive programs
- the use of geometric methods to explore the semantic space of programs

Overview

- the use of formal measures of program correctness in fitness functions
- the use of scaffolding assumptions to evolve recursive programs
- the use of geometric methods to explore the semantic space of programs

Confidence

- Ways to be confident that a system works:
  - having confidence in the way in which it is constructed
  - testing it on a rich set of test data
  - verifying formally that it satisfies some formal constraints against a model of the desired behaviour

Formal Verification: Model Checking

- Create a formal specification of desired behaviour, in some temporal logic.
  - "Once a coin has been inserted into the machine, at some state in the future coffee must be dispensed."
  - "Once a coffee has been dispensed, another coffee cannot be dispensed until another coin has been inserted."
- There are then ways of verifying that a particular program satisfies these statements (model checking; MC); essentially, a rigorous way of counting sets of paths through the program, quotiented out by each logical statement.

Using MC in GP

- Simple fitness function:
  - fitness is defined as a count of how many statements in the specification are satisfied
- Experiments:
  - some success for simple examples
  - in particular, by accumulating programs from simple ones
  - but, reaches a complexity wall fairly quickly
- Moving forward:
  - evaluating fitness from *within* the model checking algorithm, by estimating how much of the space is covered
  - using a hybrid with traditional fitness; perhaps auto-construct test data from specifications.

The Big Picture

- How can we integrate formal models of desired program behaviour with GP? More generally, interactions between GP and formal models of knowledge/behaviour?

- We assume that the specification is accurate; but, this is a dodgy assumption.
  - Perhaps, there is a way forward where we use GP to generate programs that satisfy the specification with the aim of seeing whether the specification is broken!
  - specification-generation cycle

Overview

- the use of formal measures of program correctness in fitness functions
- the use of scaffolding assumptions to evolve recursive programs
- the use of geometric methods to explore the semantic space of programs

Recursion (1)

- Consider the problem of evolving a program to reverse a list of arbitrary length.
- Not a *long* program; here is a solution:

```
function reverse(list) {
    if (empty(list)) {
        return list;
    }
    else {
        return snoc(
            reverse(tail(list)),
                head(list));
    }
}
```

- ...so, an ideal candidate for GP, yes?

Recursion (2)

- No! Actually, rather hard...small errors get amplified in the recursion.
- Instead, create a scaffolding function:

```
function correct-reverse(list){
   if (list==[1,2,3]) return([3,2,1]);
   if (list==[2,3,1]) return([1,3,2]);
   ...
}
```

- Now, we can evolve using *correct-reverse* wherever *reverse* should be used.
- It is, by definition, correct on all of the training set.

Recursion (3)

- It works!
- And it's efficient!

Reverse

(i) recursive approach	*P(M,i)*	*min I(M,i,z)*
crossover-only	0%	–
mutation-only	7%	5,984,000
crossover-and-mutation	25%	1,623,500

(i) scaffolding-based approach	*P(M,i)*	*min I(M,i,z)*
crossover-only	1%	2,299,590
mutation-only	32%	792,000
crossover-and-mutation	**69%**	**252,000**

Insert

(i) recursive approach	*P(M,i)*	*min I(M,i,z)*
crossover-only	0%	–
mutation-only	89%	201,000
crossover-and-mutation	71%	362,500

(i) scaffolding-based approach	*P(M,i)*	*min I(M,i,z)*
crossover-only	5%	228,456
mutation-only	65%	169,500
crossover-and-mutation	**100%**	**49,000**

The Big Picture

- Almost the opposite of before:
  - Rather than taking something that is usually specified by data and replacing it with a formal calcuation (MC replacing testing) here we are taking something that is usually done by calculation and replacing it by data!)
- The bigger picture here is about "intermediate evaluation"—what can we say about a program before it is completely evaluated? How can we "fill in the gaps" for something that isn't yet executable?

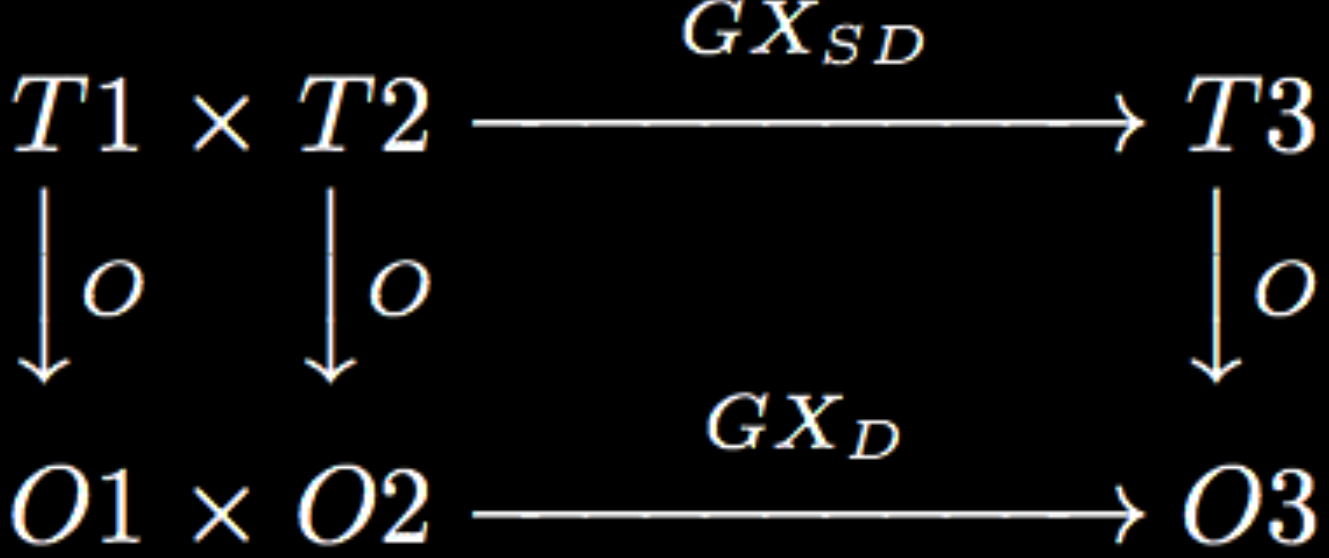
Overview

- the use of formal measures of program correctness in fitness functions
- the use of scaffolding assumptions to evolve recursive programs
- the use of geometric methods to explore the semantic space of programs

Semantics in Genetic Programming

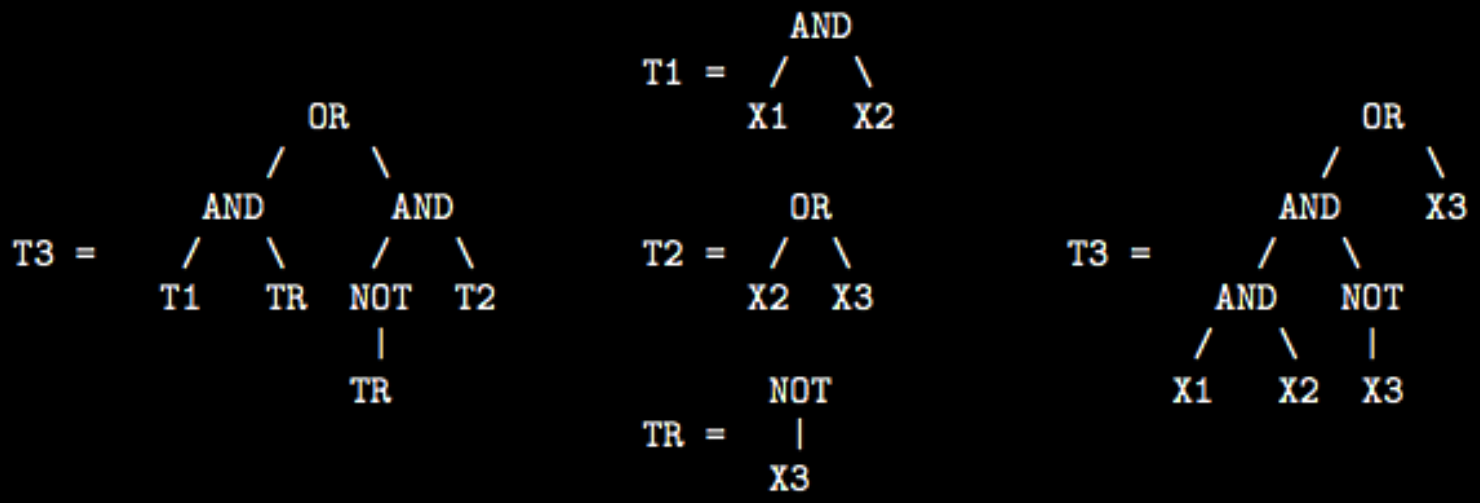
- Focusing on exploring the space of *program meanings* rather than the space of *program text*?
- The "semantics" of a program are, in this context, its set of input-output behaviours across its input space
  - either formalised in some way
  - or via a sample of inputs ("sampling semantics")
- Operators acting on the meaning of programs:
  - semantic mutation: a small change to the input-output behaviour of programs
  - semantic crossover: taking two programs and finding a new program that exhibits input-output behaviours "inbetween" the two programs

Geometric Semantic Genetic Programming: Lifting



T1, T2, T3: program text
O1, O2, O3: input-output mapping
O: genotype-phenotype mapping
GX: geometric crossover

Example: Geometric Crossover

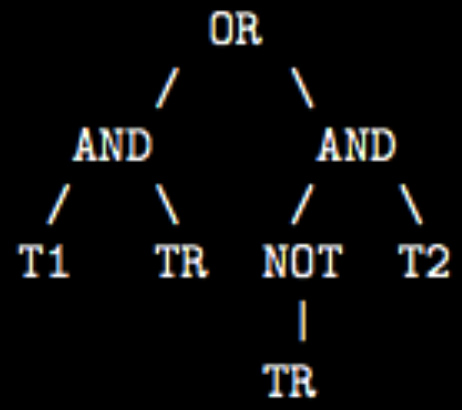


- Essentially a "random mask", selecting from the two input trees.
- Similar structures exist for other domains e.g. in curve fitting

...but hang on a minute...

- Let's have a look at this again...
- ...each time we apply it, we *double* the size of the tree...
- ...so after many generations, we have huge trees.

T3 =



- We have dealt with this via simplification...
- ...but, more recently Vanesschi et al. have noted that this can be handled by pointer manipulation; and applied these ideas to real-world medical problems.

Some Results

Problem	Hits %								Length			
	GP		GPt		SSHC		SGP					
	avg	sd	avg	sd	avg	sd	avg	sd	GP	GPt	SSHC	SGP
Comparator6	80.2	3.8	90.9	3.5	99.8	0.5	99.5	0.7	1.0	2.0	2.9	2.8
Comparator8	80.3	2.8	94.9	2.4	100.0	0.0	99.9	0.2	1.0	2.3	2.9	3.0
Comparator10	82.3	4.3	95.3	0.9	100.0	0.0	100.0	0.1	1.6	2.4	2.7	3.0
Multiplexer6	70.8	3.3	94.7	5.8	99.8	0.5	99.5	0.8	1.1	2.2	2.7	2.9
Multiplexer11	76.4	7.9	88.8	3.4	100.0	0.0	99.9	0.1	2.2	2.4	2.9	2.6
Parity5	52.9	2.4	56.3	4.9	99.7	0.9	98.1	2.1	1.4	1.7	2.9	2.9
Parity6	50.5	0.7	55.4	5.1	99.7	0.6	98.8	1.7	1.0	1.9	3.0	3.0
Parity7	50.1	0.2	51.7	2.8	99.9	0.2	99.5	0.6	1.0	1.7	3.0	3.1
Parity8	50.1	0.2	50.6	0.9	100.0	0.0	99.7	0.3	1.0	1.6	3.4	3.4
Parity9	50.0	0.0	50.2	0.1	100.0	0.0	99.5	0.3	1.0	1.3	3.8	3.8
Parity10	50.0	0.0	50.0	0.0	100.0	0.0	99.4	0.2	0.9	1.2	4.1	4.1
Random5	82.2	6.6	90.9	6.0	99.5	1.2	98.8	2.1	0.9	1.6	2.7	2.8
Random6	83.6	6.6	93.0	4.1	99.9	0.4	99.2	1.3	1.2	1.9	2.9	2.8
Random7	85.1	5.3	92.9	3.8	99.9	0.2	99.8	0.4	1.1	2.0	2.8	2.9
Random8	89.6	5.3	93.7	2.4	100.0	0.1	99.9	0.2	1.4	2.0	3.0	2.9
Random9	93.1	3.7	95.4	2.3	100.0	0.1	100.0	0.1	1.5	1.8	2.9	2.9
Random10	95.3	2.3	96.2	2.0	100.0	0.0	100.0	0.0	1.5	1.8	2.8	3.0
Random11	96.6	1.6	97.3	1.5	100.0	0.0	100.0	0.0	1.6	1.7	2.7	3.1
True5	100.0	0.0	100.0	0.0	99.9	0.6	100.0	0.0	1.1	1.3	2.0	2.4
True6	100.0	0.0	100.0	0.0	99.8	0.6	100.0	0.0	1.2	1.2	2.6	2.5
True7	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	1.2	1.2	2.9	2.6
True8	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.1	1.2	1.4	3.3	2.9

Some Results

Problem	Hits %								Length			
	GP		GPt		SSHC		SGP					
	avg	sd	avg	sd	avg	sd	avg	sd	GP	GPt	SSHC	SGP
Comparator6	80.2	3.8	90.9	3.5	99.8	0.5	99.5	0.7	1.0	2.0	2.9	2.8
Comparator8	80.3	2.8	94.9	2.4	100.0	0.0	99.9	0.2	1.0	2.3	2.9	3.0
Comparator10	82.3	4.3	95.3	0.9	100.0	0.0	100.0	0.1	1.6	2.4	2.7	3.0
Multiplexer6	70.8	3.3	94.7	5.8	99.8	0.5	99.5	0.8	1.1	2.2	2.7	2.9
Multiplexer11	76.4	7.9	88.8	3.4	100.0	0.0	99.9	0.1	2.2	2.4	2.9	2.6
Parity5	52.9	2.4	56.3	4.9	99.7	0.9	98.1	2.1	1.4	1.7	2.9	2.9
Parity6	50.5	0.7	55.4	5.1	99.7	0.6	98.8	1.7	1.0	1.9	3.0	3.0
Parity7	50.1	0.2	51.7	2.8	99.9	0.2	99.5	0.6	1.0	1.7	3.0	3.1
Parity8	50.1	0.2	50.6	0.9	100.0	0.0	99.7	0.3	1.0	1.6	3.4	3.4
Parity9	50.0	0.0	50.2	0.1	100.0	0.0	99.5	0.3	1.0	1.3	3.8	3.8
Parity10	50.0	0.0	50.0	0.0	100.0	0.0	99.4	0.2	0.9	1.2	4.1	4.1
Random5	82.2	6.6	90.9	6.0	99.5	1.2	98.8	2.1	0.9	1.6	2.7	2.8
Random6	83.6	6.6	93.0	4.1	99.9	0.4	99.2	1.3	1.2	1.9	2.9	2.8
Random7	85.1	5.3	92.9	3.8	99.9	0.2	99.8	0.4	1.1	2.0	2.8	2.9
Random8	89.6	5.3	93.7	2.4	100.0	0.1	99.9	0.2	1.4	2.0	3.0	2.9
Random9	93.1	3.7	95.4	2.3	100.0	0.1	100.0	0.1	1.5	1.8	2.9	2.9
Random10	95.3	2.3	96.2	2.0	100.0	0.0	100.0	0.0	1.5	1.8	2.8	3.0
Random11	96.6	1.6	97.3	1.5	100.0	0.0	100.0	0.0	1.6	1.7	2.7	3.1
True5	100.0	0.0	100.0	0.0	99.9	0.6	100.0	0.0	1.1	1.3	2.0	2.4
True6	100.0	0.0	100.0	0.0	99.8	0.6	100.0	0.0	1.2	1.2	2.6	2.5
True7	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	1.2	1.2	2.9	2.6
True8	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.1	1.2	1.4	3.3	2.9

Some Results

Problem	Hits %								Length			
	GP		GPt		SSHC		SGP					
	avg	sd	avg	sd	avg	sd	avg	sd	GP	GPt	SSHC	SGP
Comparator6	80.2	3.8	90.9	3.5	99.8	0.5	99.5	0.7	1.0	2.0	2.9	2.8
Comparator8	80.3	2.8	94.9	2.4	100.0	0.0	99.9	0.2	1.0	2.3	2.9	3.0
Comparator10	82.3	4.3	95.3	0.9	100.0	0.0	100.0	0.1	1.6	2.4	2.7	3.0
Multiplexer6	70.8	3.3	94.7	5.8	99.8	0.5	99.5	0.8	1.1	2.2	2.7	2.9
Multiplexer11	76.4	7.9	88.8	3.4	100.0	0.0	99.9	0.1	2.2	2.4	2.9	2.6
Parity5	52.9	2.4	56.3	4.9	99.7	0.9	98.1	2.1	1.4	1.7	2.9	2.9
Parity6	50.5	0.7	55.4	5.1	99.7	0.6	98.8	1.7	1.0	1.9	3.0	3.0
Parity7	50.1	0.2	51.7	2.8	99.9	0.2	99.5	0.6	1.0	1.7	3.0	3.1
Parity8	50.1	0.2	50.6	0.9	100.0	0.0	99.7	0.3	1.0	1.6	3.4	3.4
Parity9	50.0	0.0	50.2	0.1	100.0	0.0	99.5	0.3	1.0	1.3	3.8	3.8
Parity10	50.0	0.0	50.0	0.0	100.0	0.0	99.4	0.2	0.9	1.2	4.1	4.1
Random5	82.2	6.6	90.9	6.0	99.5	1.2	98.8	2.1	0.9	1.6	2.7	2.8
Random6	83.6	6.6	93.0	4.1	99.9	0.4	99.2	1.3	1.2	1.9	2.9	2.8
Random7	85.1	5.3	92.9	3.8	99.9	0.2	99.8	0.4	1.1	2.0	2.8	2.9
Random8	89.6	5.3	93.7	2.4	100.0	0.1	99.9	0.2	1.4	2.0	3.0	2.9
Random9	93.1	3.7	95.4	2.3	100.0	0.1	100.0	0.1	1.5	1.8	2.9	2.9
Random10	95.3	2.3	96.2	2.0	100.0	0.0	100.0	0.0	1.5	1.8	2.8	3.0
Random11	96.6	1.6	97.3	1.5	100.0	0.0	100.0	0.0	1.6	1.7	2.7	3.1
True5	100.0	0.0	100.0	0.0	99.9	0.6	100.0	0.0	1.1	1.3	2.0	2.4
True6	100.0	0.0	100.0	0.0	99.8	0.6	100.0	0.0	1.2	1.2	2.6	2.5
True7	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	1.2	1.2	2.9	2.6
True8	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.1	1.2	1.4	3.3	2.9

Some Results

Problem	Hits %								Length			
	GP		GPt		SSHC		SGP					
	avg	sd	avg	sd	avg	sd	avg	sd	GP	GPt	SSHC	SGP
Comparator6	80.2	3.8	90.9	3.5	99.8	0.5	99.5	0.7	1.0	2.0	2.9	2.8
Comparator8	80.3	2.8	94.9	2.4	100.0	0.0	99.9	0.2	1.0	2.3	2.9	3.0
Comparator10	82.3	4.3	95.3	0.9	100.0	0.0	100.0	0.1	1.6	2.4	2.7	3.0
Multiplexer6	70.8	3.3	94.7	5.8	99.8	0.5	99.5	0.8	1.1	2.2	2.7	2.9
Multiplexer11	76.4	7.9	88.8	3.4	100.0	0.0	99.9	0.1	2.2	2.4	2.9	2.6
Parity5	52.9	2.4	56.3	4.9	99.7	0.9	98.1	2.1	1.4	1.7	2.9	2.9
Parity6	50.5	0.7	55.4	5.1	99.7	0.6	98.8	1.7	1.0	1.9	3.0	3.0
Parity7	50.1	0.2	51.7	2.8	99.9	0.2	99.5	0.6	1.0	1.7	3.0	3.1
Parity8	50.1	0.2	50.6	0.9	100.0	0.0	99.7	0.3	1.0	1.6	3.4	3.4
Parity9	50.0	0.0	50.2	0.1	100.0	0.0	99.5	0.3	1.0	1.3	3.8	3.8
Parity10	50.0	0.0	50.0	0.0	100.0	0.0	99.4	0.2	0.9	1.2	4.1	4.1
Random5	82.2	6.6	90.9	6.0	99.5	1.2	98.8	2.1	0.9	1.6	2.7	2.8
Random6	83.6	6.6	93.0	4.1	99.9	0.4	99.2	1.3	1.2	1.9	2.9	2.8
Random7	85.1	5.3	92.9	3.8	99.9	0.2	99.8	0.4	1.1	2.0	2.8	2.9
Random8	89.6	5.3	93.7	2.4	100.0	0.1	99.9	0.2	1.4	2.0	3.0	2.9
Random9	93.1	3.7	95.4	2.3	100.0	0.1	100.0	0.1	1.5	1.8	2.9	2.9
Random10	95.3	2.3	96.2	2.0	100.0	0.0	100.0	0.0	1.5	1.8	2.8	3.0
Random11	96.6	1.6	97.3	1.5	100.0	0.0	100.0	0.0	1.6	1.7	2.7	3.1
True5	100.0	0.0	100.0	0.0	99.9	0.6	100.0	0.0	1.1	1.3	2.0	2.4
True6	100.0	0.0	100.0	0.0	99.8	0.6	100.0	0.0	1.2	1.2	2.6	2.5
True7	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	1.2	1.2	2.9	2.6
True8	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.1	1.2	1.4	3.3	2.9

The Big Picture

- Working in the space of the meaning of programs rather than program text
- (but, text is still needed)
- *Representation* matters more than *search algorithm*.

- Is this bag-of-tricks useful elsewhere (e.g. in machine learning)?

Thanks to

- Alberto Moraglio
- Krzysztof Krawiec
- Fernando Otero
- Alex Freitas
- Simon Thompson
- Lawrence Beadle
- He Pei

Questions/Comments