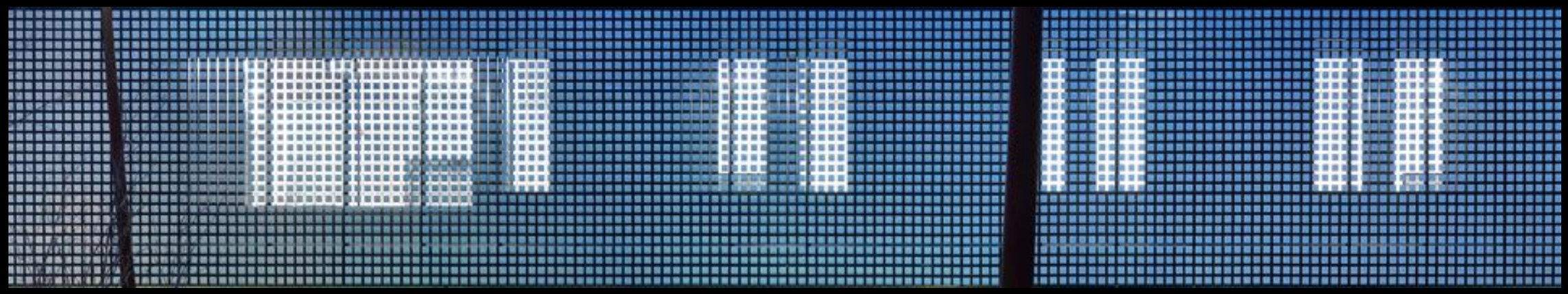


fitness as task-relevant information accumulation

colin johnson / university of kent

Accumulating Information

- Programs execute one step at a time.
- Correct programs accumulate some information at each of those steps.
- How can we quantify that information gain?
- One way is through *information distances:* algorithms that measure how easy it is to transform one thing into another.

Normalised Compression Distance

- Vitányi and colleagues have invented Normalised Compression Distances.
- Compares the compressibility of two sequences *x* and *y* w.r.t a compressor *Z*.

$$NCD_Z(x,y) = \frac{Z(xy) - \min(Z(x), Z(y))}{\max(Z(x), Z(y))}$$

Function	TC!	TC2	TC3	TC4	TC5	TC6	TC7	TC8…
v1	0	0	0	0	0	0	0	0
v2	0	0	0	0	1	1	1	1
v3	0	0	1	1	0	0	1	1
v4	0	1	0	1	0	1	0	1
f1 = v1 XOR v2	0	0	0	0	1	1	1	1
f2 = v3 XOR v4	0	1	1	0	0	1	1	0
f3 = f1 XOR f2	0	1	1	0	1	0	0	1
TARGET	0	1	1	0	1	0	0	1

Function	NCD to target
v1	0.290
v2	0.258
v3	0.290
v4	0.323
f1 = v1 XOR v2	0.194
f2 = v3 XOR v4	0.258
f3 = f1 XOR f2	0.129

Function	NCD to target
v1	0.290
v2	0.258
v3	0.290
v4	0.323
f1 = v1 XOR v2	0.194
f2 = v3 XOR v4	0.258
f3 = f1 XOR f2	0.129

Function	NCD to target
v1	0.290
v2	0.258
v3	0.290
v4	0.323
f1 = v1 XOR v2	0.194
f2 = v3 XOR v4	0.258
f3 = f1 XOR f2	0.129

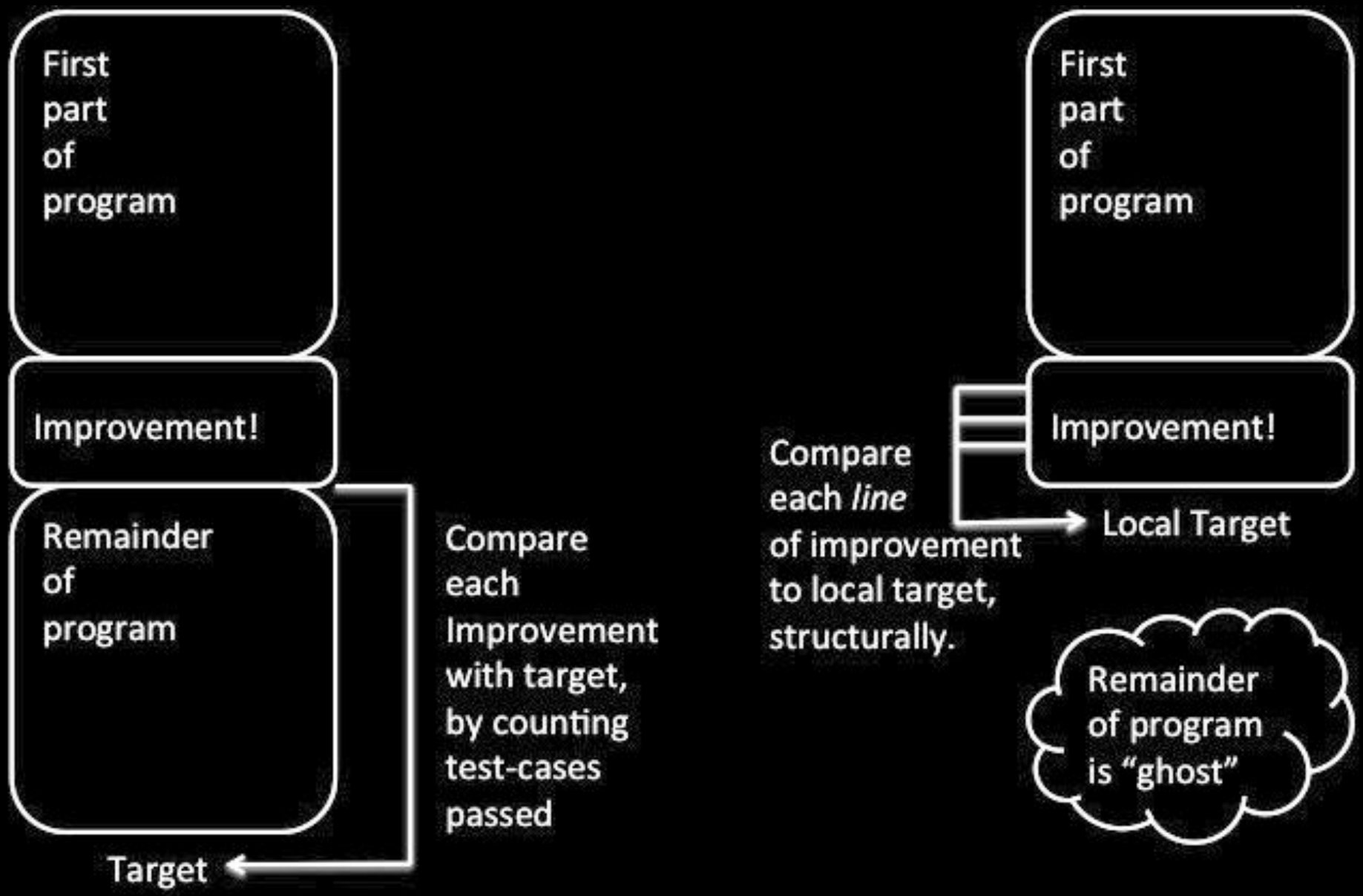
Function	NCD to target
v1	0.290
v2	0.258
v3	0.290
v4	0.323
f1 = v1 XOR v2	0.194
f2 = v3 XOR v4	0.258
f3 = f1 OR f2	0.226

Function	NCD to target
v1	0.290
v2	0.258
v3	0.290
v4	0.323
f1 = v1 XOR v1	0.323
f2 = v3 XOR v4	0.258
f3 = f1 XOR f2	0.129

Measuring Progress

- By measuring the NCD, we are measuring how much computational progress has been made to the target.
- Not by using the rest of the code but by *measuring* the information gained direcly.
- Note importantly that these numbers are generated step-by-step as the program executes.

Information Gain and GI



fitNESS vs. FITness

- Traditional GI/GP has focused on what we might call fitNESS, that is, measuring the amount of things that are fit. By contrast, we advocate the use of FITness, i.e. the amount by which the outputs from the code fit (structurally) with the target output.

Potential Advantages

- Firstly, it does not rely on the correctness of the remainder of the program.
- Secondly, it puts pressure on the improvement to make improvements that solve coherent sub-problems.
- Thirdly, each line of the improvement can be learned one-by-one; we do not need to have a whole improvement before we can evalu- ate the quality of a line.