GECCO 2018 - The Genetic and Evolutionary Computation Conference
Kyoto, Japan

A Novel Fitness Function for Automated Program Repair Based on Source Code Checkpoints

Eduardo Souza
Instituto de Informática
Universidade Federal de Goiás
Brazil

Claire Le Goues
School of Computer Science
Carnegie Mellon University
USA

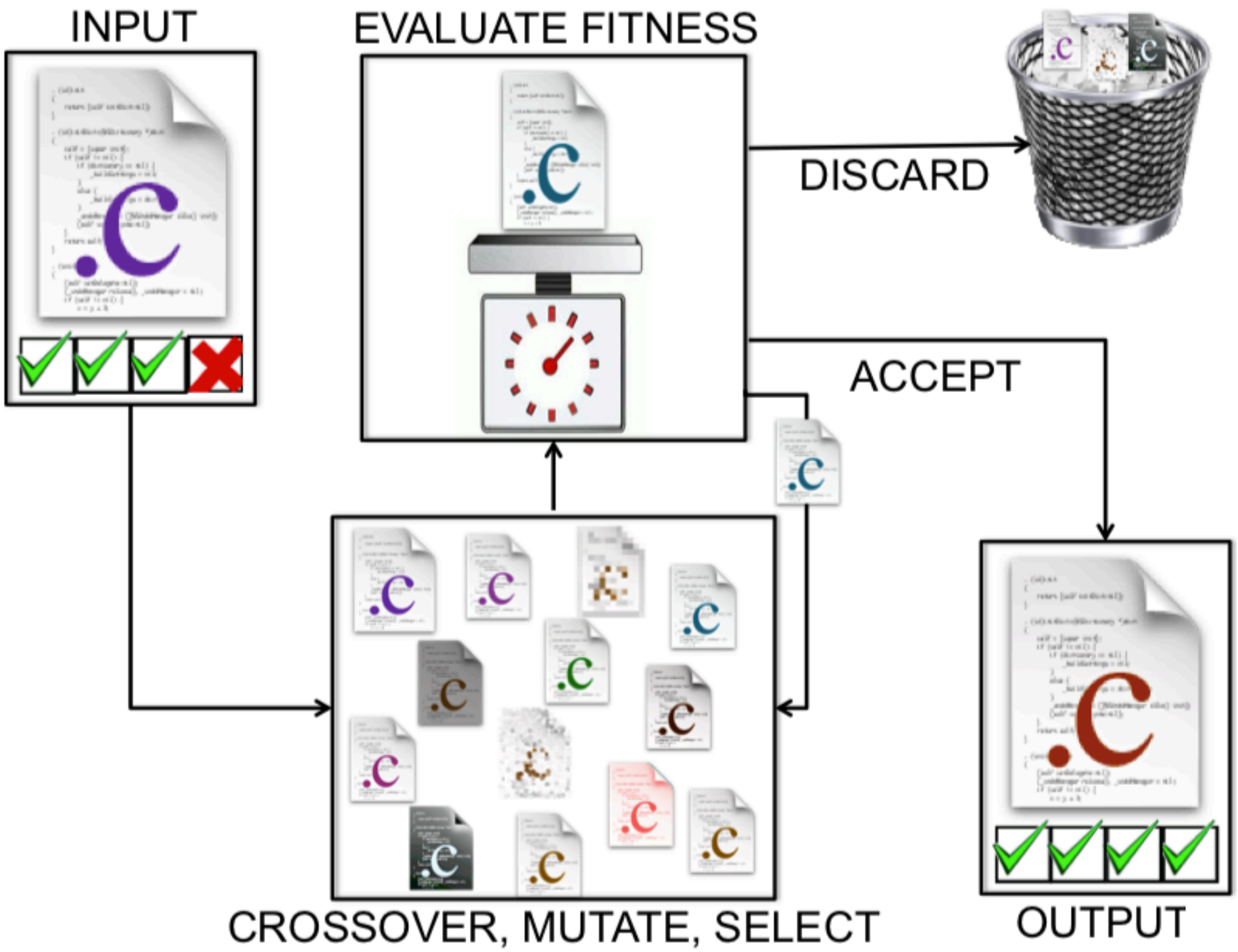
Celso Camilo-Junior
Instituto de Informática
Universidade Federal de Goiás
Brazil

UFG UNIVERSIDADE FEDERAL DE GOIÁS I4Soft Intelligence for Software *in collaboration with* Carnegie Mellon University squaresLab Software Quality in Real Evolving Systems

Bugs

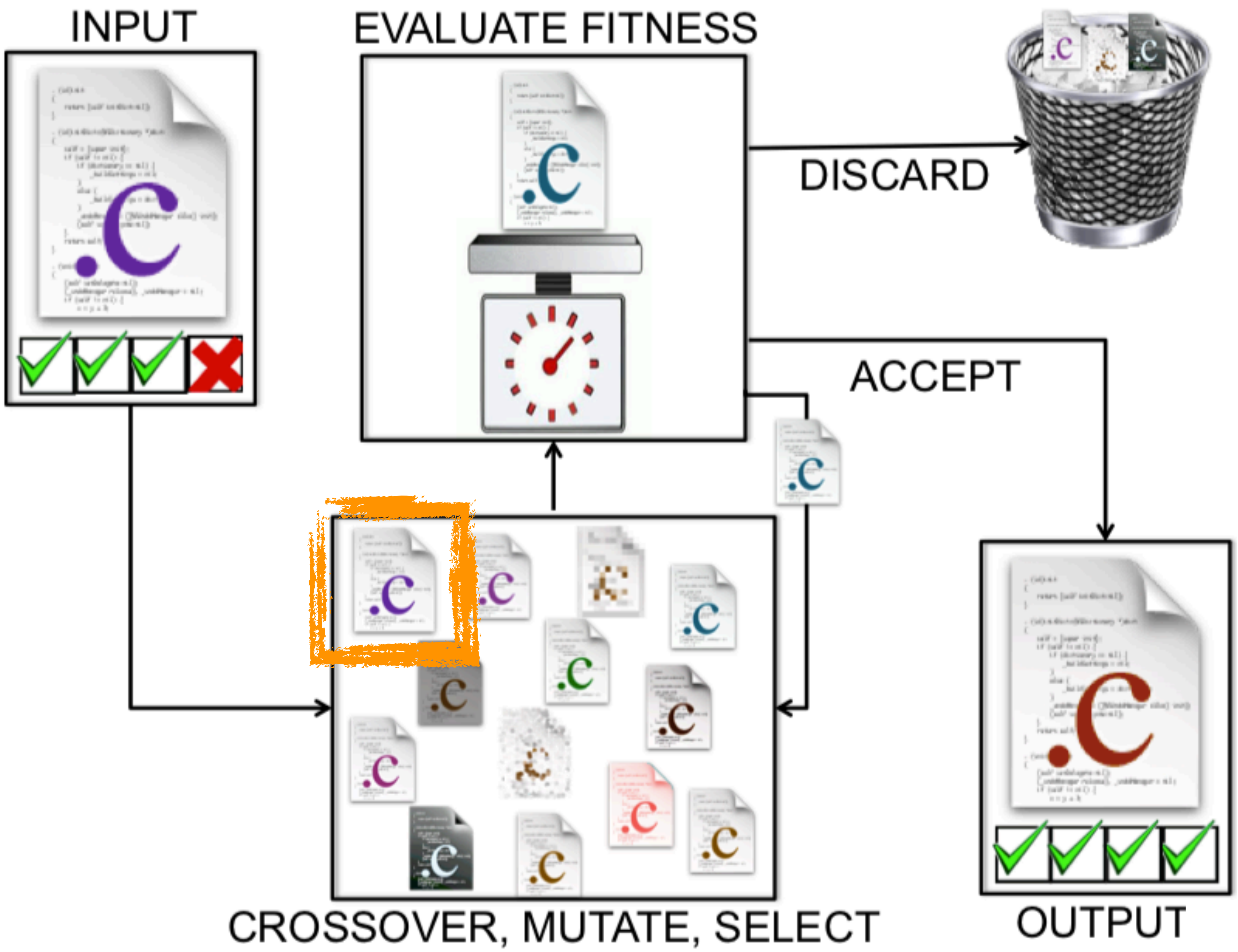
Automated Program Repair

GenProg



Source: Claire Le Goues et. al.

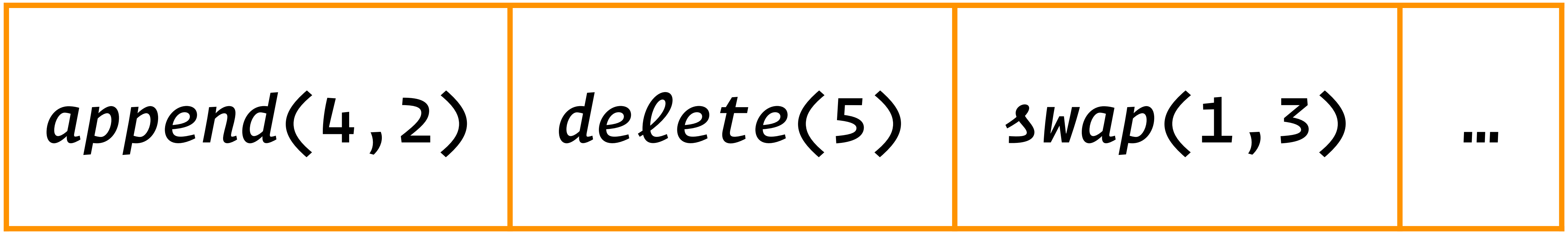
GenProg



Source: Claire Le Goues et. al.

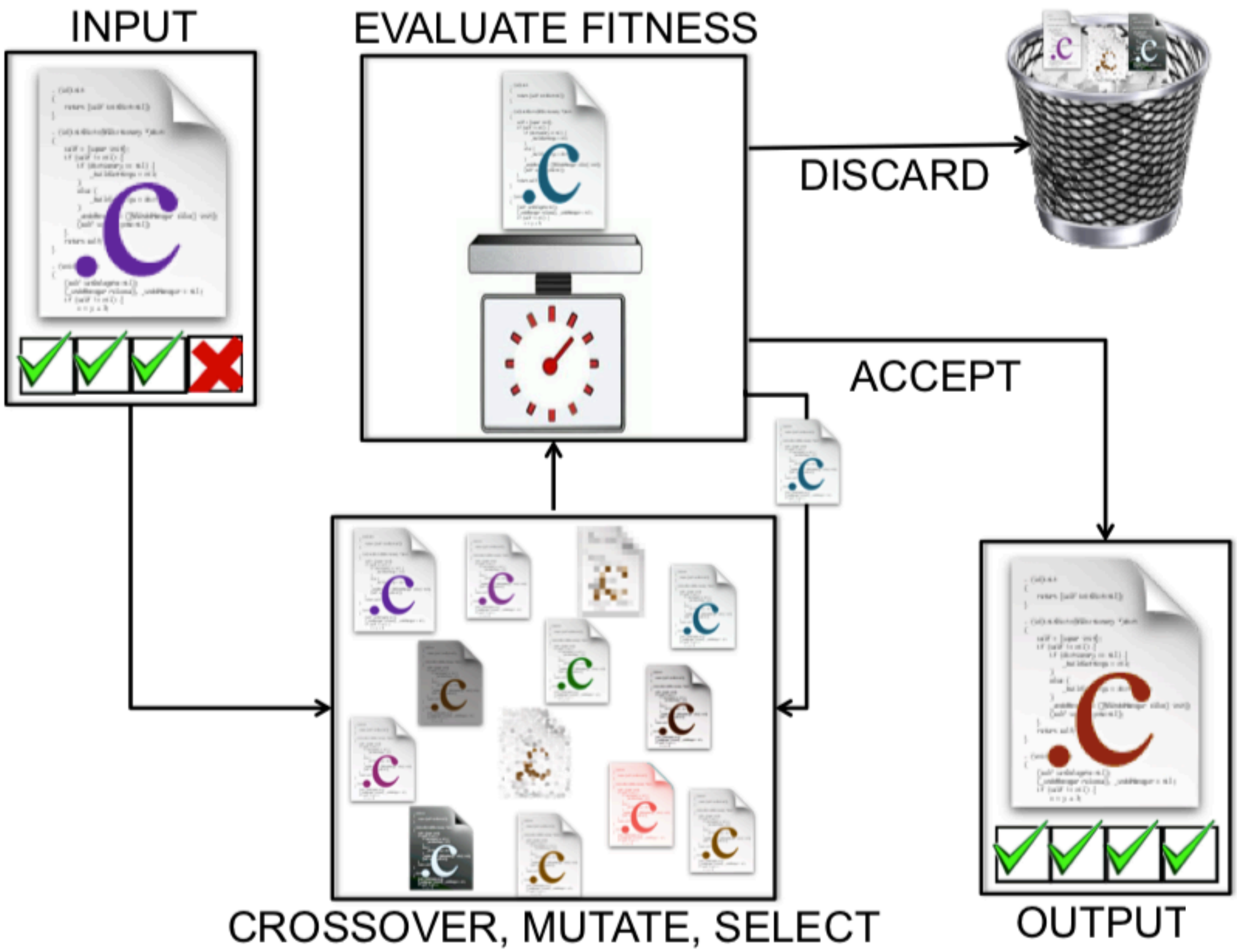
GenProg

Genotypic Representation



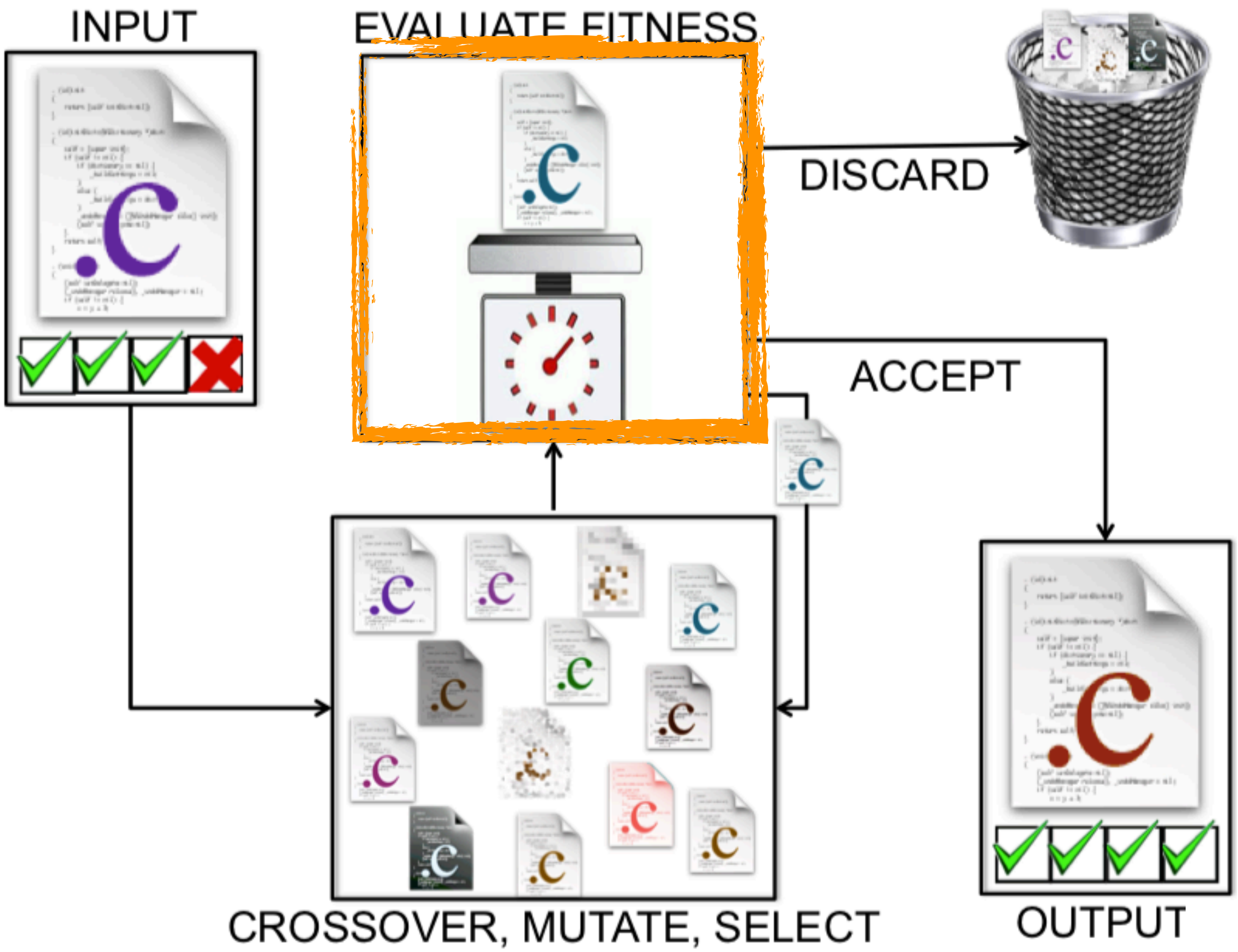
append(4,2) | *delete(5)* | *swap(1,3)* | …

GenProg



Source: Claire Le Goues et. al.

GenProg



Source: Claire Le Goues et. al.

GenProg

Canonical Fitness

$$fit(ind) = \sum_{tc \in TS} type(tc) \times pass(tc)$$

Where:

tc = Test case

TS = Test suite

type(tc) = Acts as a weight whether *tc* is $\boxed{\text{positive}}$ or $\boxed{\text{negative}}$

pass(tc) = Whether *tc* passed or not

The problem

Population

Ind#1:	append(4,2)	delete(5)	swap(1,3)
Ind#2:	delete(4)		
Ind#3:	swap(4,5)	append(1,3)	

Population

				Fitness
Ind#1:	*append(4,2)*	*delete(5)*	*swap(1,3)*	5
Ind#2:	*delete(4)*			5
Ind#3:	*swap(4,5)*	*append(1,3)*		5

GenProg

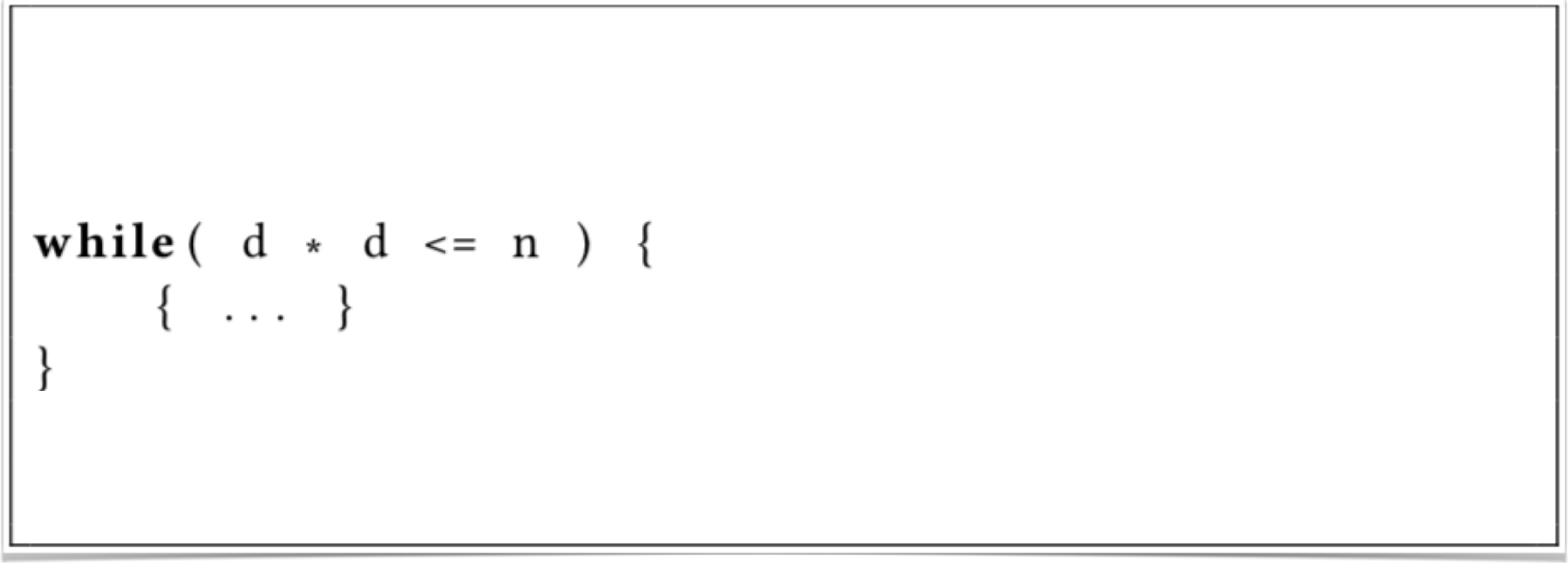
Population

				Fitness
Ind#1:	*append(4,2)*	*delete(5)*	*swap(1,3)*	5
Ind#2:	*delete(4)*			5
Ind#3:	*swap(4,5)*	*append(1,3)*		5

Plateaus!

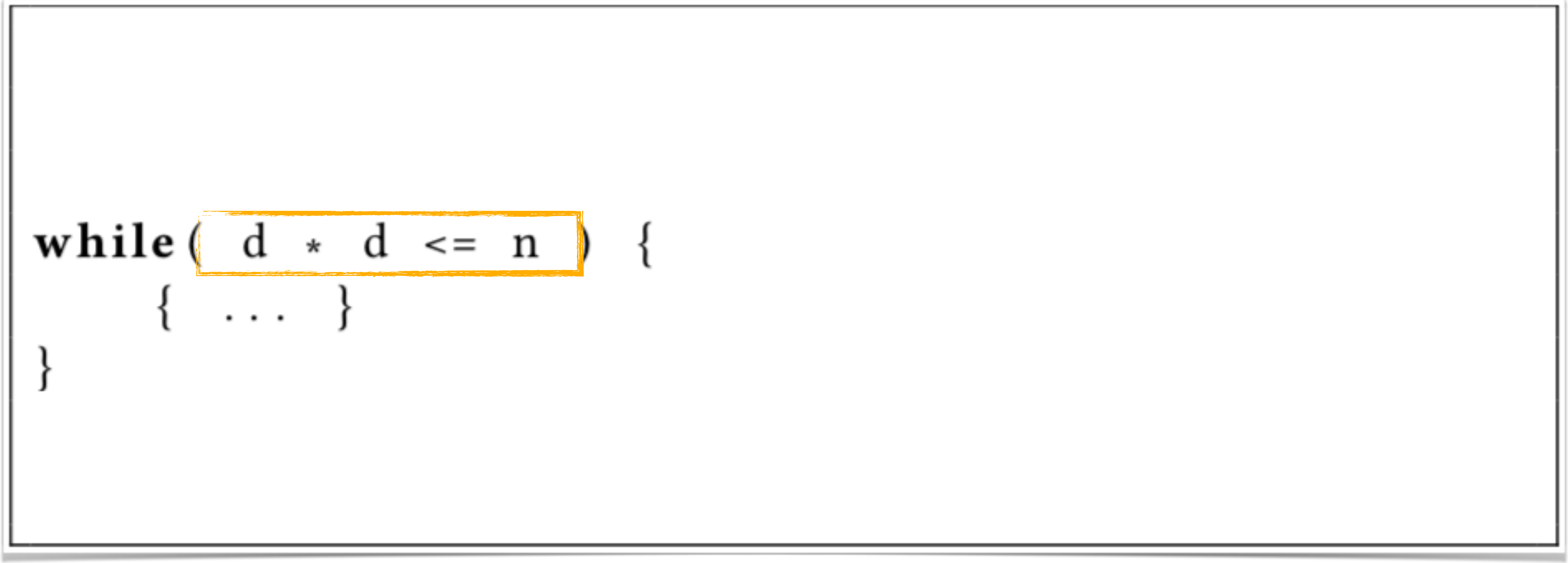
RQ1: How can we increase granularity of the fitness information?

Checkpoints



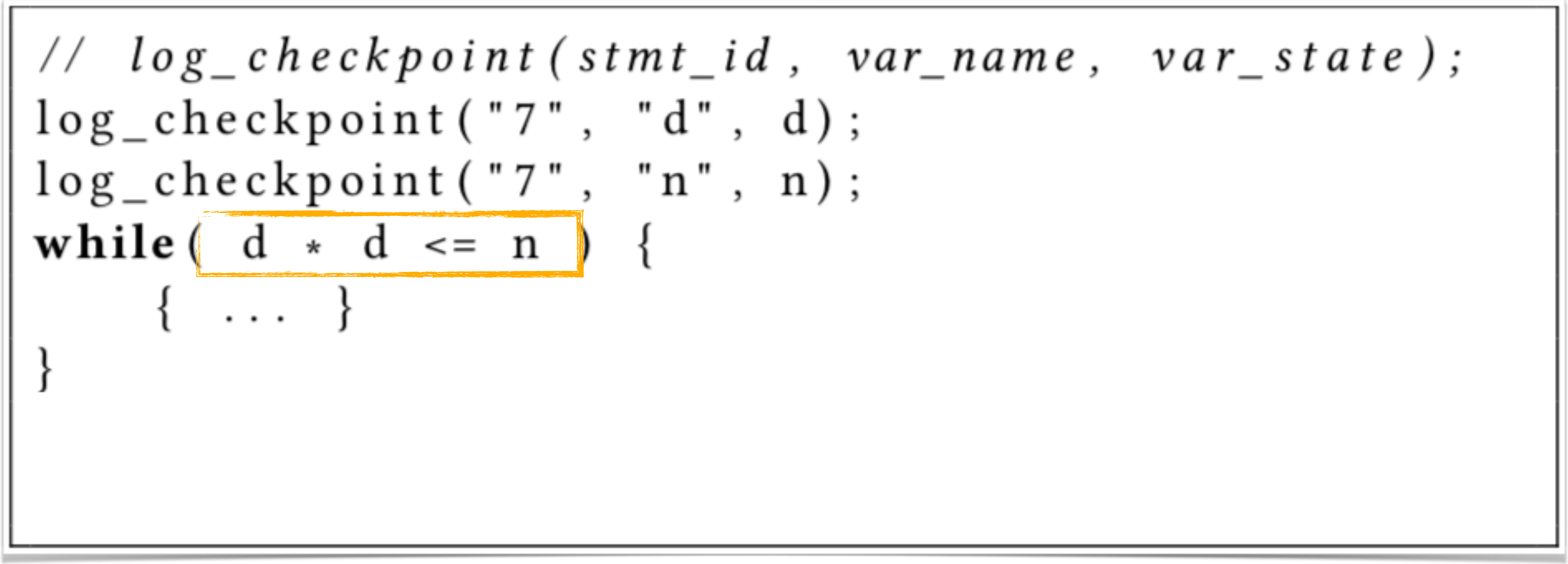
```
while( d * d <= n ) {
        { ... }
}
```

Checkpoints



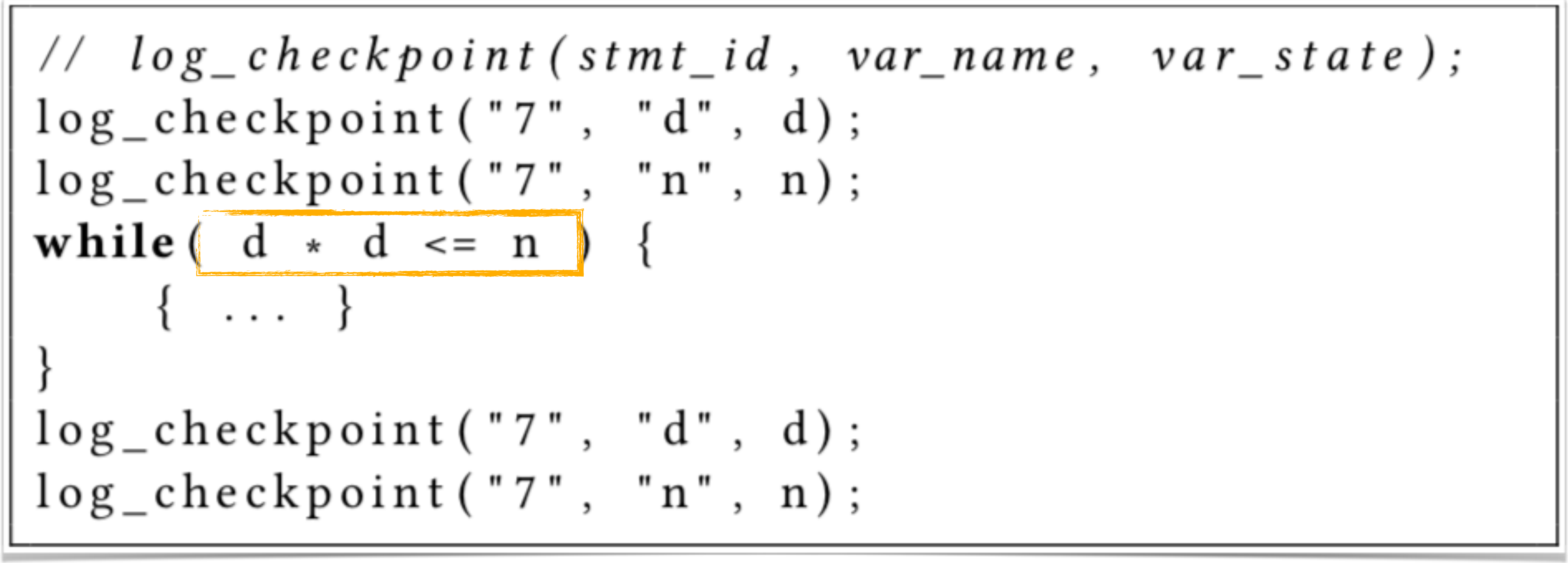
```
while ( d * d <= n ) {
        { ... }
}
```

Checkpoints



```
// log_checkpoint(stmt_id, var_name, var_state);
log_checkpoint("7", "d", d);
log_checkpoint("7", "n", n);
while( d * d <= n ) {
    { ... }
}
```

Checkpoints



```
// log_checkpoint(stmt_id, var_name, var_state);
log_checkpoint("7", "d", d);
log_checkpoint("7", "n", n);
while( d * d <= n ) {
    { ... }
}
log_checkpoint("7", "d", d);
log_checkpoint("7", "n", n);
```

The Novel Fitness Function

$$0.7 \times canonical + 0.3 \times checkpoints$$

The Novel Fitness Function

$$0.7 \times canonical + 0.3 \times checkpoints$$

$$checkpoints = \frac{posScore + negScore}{|TS|}$$

The Novel Fitness Function

$$checkpoints = \frac{posScore + negScore}{|TS|}$$

$$posScore = \sum_{tc \in TS_{pos}} max\{\frac{varsNotChanged(tc)}{|trackedVars|}, pass(tc)\}$$

The Novel Fitness Function

$$checkpoints = \frac{posScore + negScore}{|TS|}$$

$$negScore = \begin{cases} \sum_{tc \in TS_{neg}} pass(tc), & \textbf{if } changes = |TS| \\ \sum_{tc \in TS_{neg}} max\{0.5 + wr, pass(tc)\}, & \textbf{if } changes \neq |TS| \end{cases}$$

The Novel Fitness Function

$$checkpoints = \frac{posScore + negScore}{|TS|}$$

$$negScore = \begin{cases} \sum_{tc \in TS_{neg}} pass(tc), & \textbf{if } changes = |TS| \\ \sum_{tc \in TS_{neg}} max\{0.5 + wr, pass(tc)\}, & \textbf{if } changes \neq |TS| \end{cases}$$

The Novel Fitness Function

$$checkpoints = \frac{posScore + negScore}{|TS|}$$

$$negScore = \begin{cases} \sum\limits_{tc \in TS_{neg}} pass(tc), & \textbf{if } changes = |TS| \\ \sum\limits_{tc \in TS_{neg}} max\{0.5 + wr, pass(tc)\}, & \textbf{if } changes \neq |TS| \end{cases}$$

$$changes = \sum_{tc \in TS_{neg}} \frac{varsChanged(tc)}{|trackedVars|}$$

The Novel Fitness Function

$$checkpoints = \frac{posScore + negScore}{|TS|}$$

$$negScore = \begin{cases} \sum_{tc \in TS_{neg}} pass(tc), & \textbf{if } changes = |TS| \\ \sum_{tc \in TS_{neg}} max\{0.5 + wr, pass(tc)\}, & \textbf{if } changes \neq |TS| \end{cases}$$

$$changes = \sum_{tc \in TS_{neg}} \frac{varsChanged(tc)}{|trackedVars|}$$

The Novel Fitness Function

$$checkpoints = \frac{posScore + negScore}{|TS|}$$

$$negScore = \begin{cases} \sum\limits_{tc\,\in\,TS_{neg}} pass(tc), & \textbf{if } changes = |TS| \\ \sum\limits_{tc\,\in\,TS_{neg}} max\{\boxed{0.5 + wr,}\, pass(tc)\}, & \textbf{if } changes \neq |TS| \end{cases}$$

$$changes = \sum_{tc\,\in\,TS_{neg}} \frac{varsChanged(tc)}{|trackedVars|}$$

The Novel Fitness Function

$$checkpoints = \frac{posScore + negScore}{|TS|}$$

$$negScore = \begin{cases} \sum_{tc \in TS_{neg}} pass(tc), & \textbf{if } changes = |TS| \\ \sum_{tc \in TS_{neg}} max\{\boxed{0.5 + wr}, pass(tc)\}, & \textbf{if } changes \neq |TS| \end{cases}$$

$$changes = \sum_{tc \in TS_{neg}} \frac{varsChanged(tc)}{|trackedVars|} \qquad wr = 0.4 \times \frac{changesInFaultyStmts}{|faultyStmts|}$$

RQ2: Can we improve expressiveness and efficiency of GenProg?

Experiments

Benchmark	Program	LOC	Bugs	Description
IntroClass	Checksum	13	19	checksum for a string
	Digits	15	21	digits of a number
	Median	24	25	median of 3 numbers
	Smallest	20	25	min of 4 numbers
	Syllables	23	22	count vowels
ManyBugs	Gzip	491k	5	data compression utility
	Libtiff	77k	24	image processing library
	Wireshark	2,814k	8	network packet analyzer

Search budget

- **ManyBugs**
  - 10 executions
  - 10 generations
  - 40 individuals
  - 4 elitists

- **IntroClass**
  - 20 executions
  - 30 generations
  - 40 individuals
  - 4 elitists

RQ1: How can we increase the granularity of the fitness information?

Benchmark	Program	Canonical		Checkpoints		*p*-value
		Avg	StdDev	Avg	StdDev	
IntroClass	Checksum	96.55%	15.46%	95.75%	17.46%	0.1815
	Digits	91.75%	20.48%	**90.23%** ↓	22.23% ↑	0.0026
	Median	94.52%	18.26%	**89.3%** ↓	23.42% ↑	$p < 0.001$
	Smallest	76.73%	32.46%	**61.85%** ↓	34.47% ↑	$p < 0.001$
	Syllables	98.13%	8.58%	**92.88%** ↓	15.07% ↑	$p < 0.001$
ManyBugs	Gzip	99.16%	2.57%	**80.43%** ↓	19.05% ↑	$p < 0.001$
	Libtiff	78.26%	30.65%	73.05%	31.31%	0.0752
	Wireshark	93.62%	16.73%	**80.91%** ↓	19.81% ↑	$p < 0.001$

Experiments

RQ2: Can we improve expressiveness and efficiency of GenProg?

Benchmark	Program	Canonical fixes			Checkpoints fixes		
		Bugs fixed	Runs w. fix	Avg. evals.	Bugs fixed	Runs w. fix	Avg. evals.
IntroClass	Checksum	2	27	240.37	**3 ↑50%**	**32 ↑18%**	**60.68 ↓3.9x**
	Digits	7	55	189.63	**8 ↑14%**	**79 ↑43%**	**93.93 ↓2.0x**
	Median	6	**76**	164.68	6 =	75 ↓-1.3%	**31.13 ↓5.2x**
	Smallest	25	466	637.52	25 =	**483 ↑3.6%**	**106.56 ↓5.9x**
	Syllables	1	13	156.92	**2 ↑100%**	**23 ↑76%**	**69.39 ↓2.2x**
ManyBugs	Gzip	1	4	**54.75**	**2 ↑100%**	**10 ↑150%**	66.80 ↑0.8x
	Libtiff	17	114	2,220.65	17 =	**128 ↑12%**	**54.71 ↓40x**
	Wireshark	2	19	1,104.42	2 =	**20 ↑5.2%**	**17.5 ↓63x**

Future work

Concluding Remarks

GECCO 2018 - The Genetic and Evolutionary Computation Conference
Kyoto, Japan

A Novel Fitness Function for Automated Program Repair Based on Source Code Checkpoints

Eduardo Souza
Instituto de Informática
Universidade Federal de Goiás
Brazil

Claire Le Goues
School of Computer Science
Carnegie Mellon University
USA

Celso Camilo-Junior
Instituto de Informática
Universidade Federal de Goiás
Brazil

UFG UNIVERSIDADE FEDERAL DE GOIÁS

I4Soft Intelligence for Software

in collaboration with

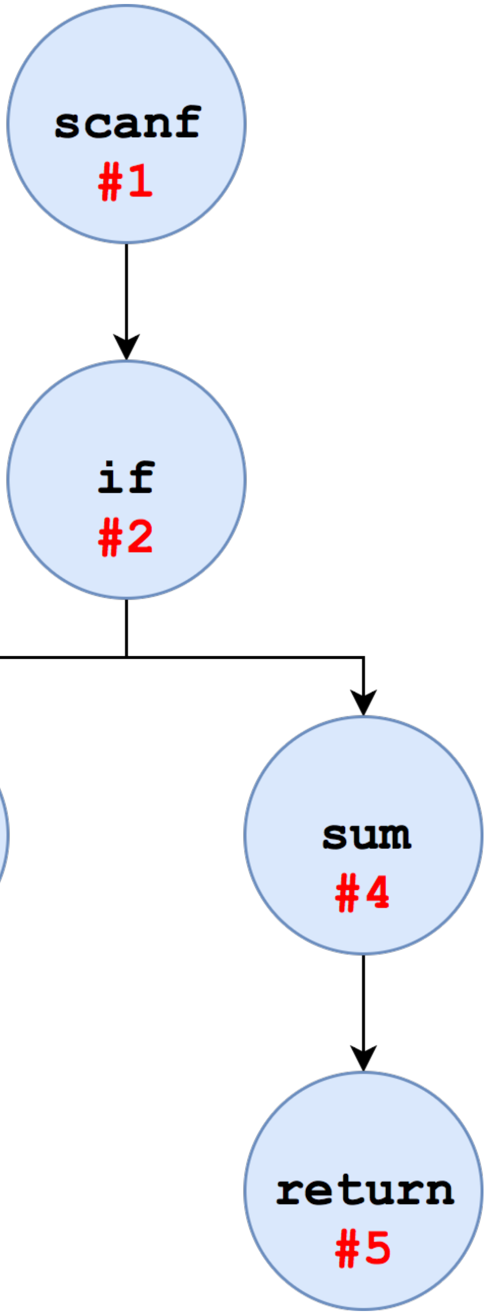
Carnegie Mellon University

squaresLab Software Quality in Real Evolving Systems

Back-up slides

GenProg

Abstract Syntax Tree



Search Space Cardinality

$$(|op| \times |wc| \times |wr|)^n$$

Where:

op = operations = $\{ \, append, \, swap, \, delete \, \}$

wc = which statement will be used

wr = where this statement will be used

n = number of edit operations that are necessary to constitute a fix

Checkpoints' metric

- Given an individual:

    1. Apply checkpoints

    2. Apply test cases

    3. Generate canonical and checkpoints′ outputs

    4. Compute **checkpoints′ metric**

    5. Produce final fitness score

Buggy Software

- **Bugs** are **prevalent** and **costly**;
- Fixing bugs is **crucial** to maintaining **software quality**;
- Bug fixing is known to be **difficult**, time-consuming, laborious, and **very expensive**.