Fuzzy Edit Sequences in Genetic Improvement

Aymeric Blot

University College London

GI@ICSE'19 — 28 May 2019

Fuzzy Edit Sequences (in GI)

"Edit Sequences" ("Patches")

- Mutant source code representation

"Semantics"

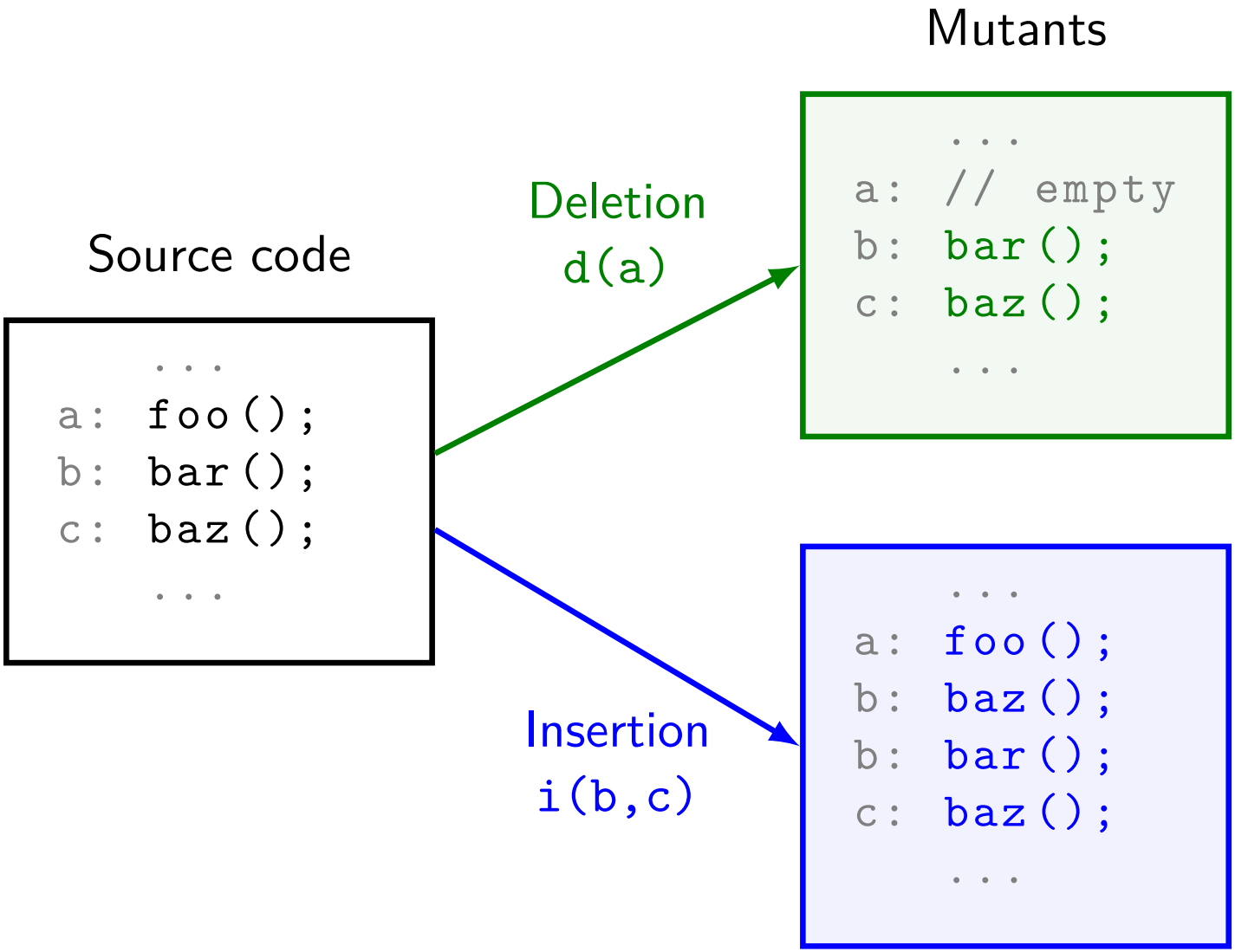
- Add significance to individual edits/mutations
- Guide the GI search process to preserve meaning

"Fuzzy"

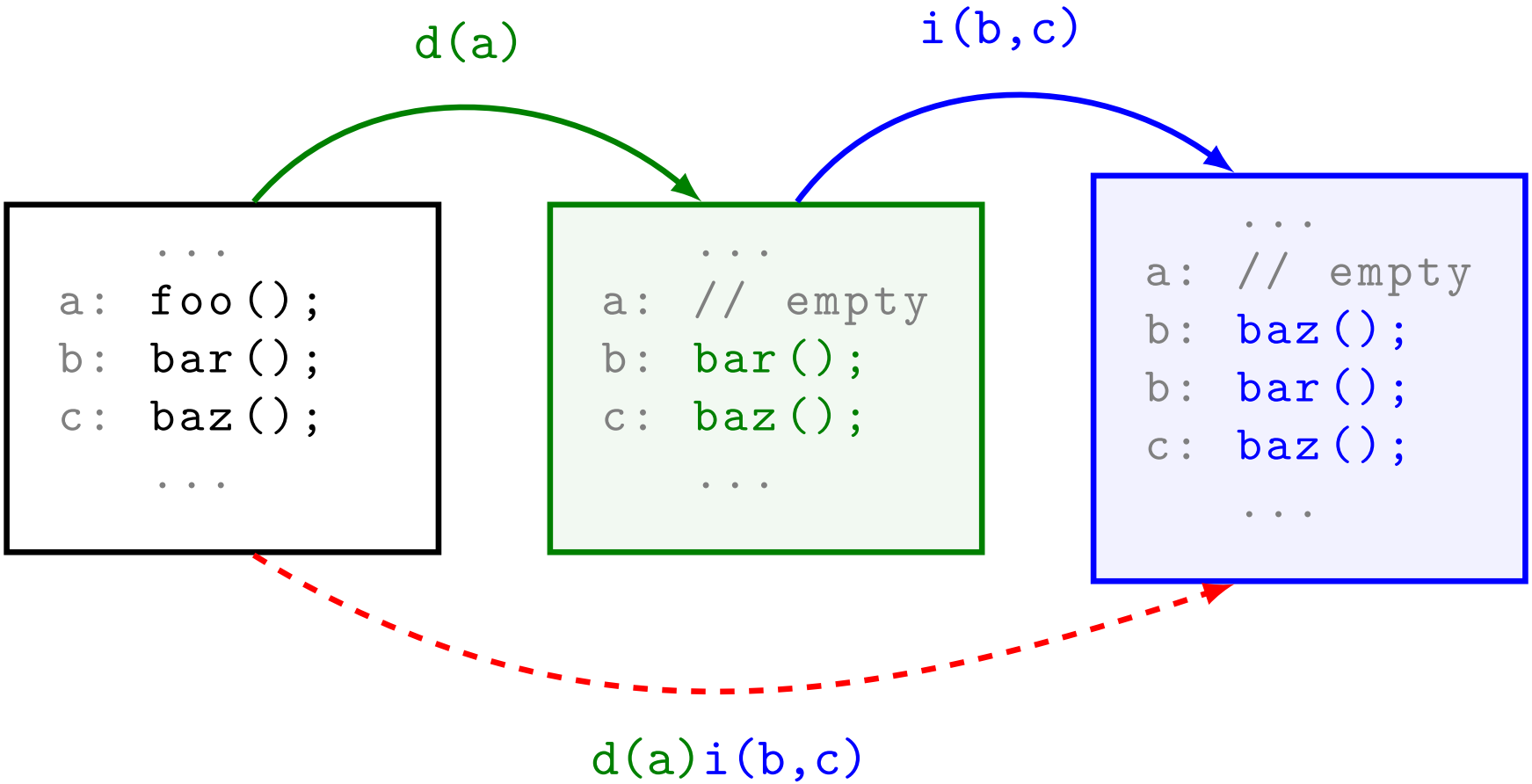
- ✖ Fuzzing
  - Fuzz testing
  - Automatic random test input generation
- ✔ Fuzzy Matching
  - Fuzzy string searching
  - Approximate string matching

Petke et al., IEEE Trans. Evol. Comput., 2018

Edits . . .


Source code
. . .
a: foo();
b: bar();
c: baz();
. . .
Deletion
d(a)
Insertion
i(b,c)
Mutants
. . .
a: // empty
b: bar();
c: baz();
. . .
. . .
a: foo();
b: baz();
b: bar();
c: baz();
. . .

Edit Sequences . . .



Langdon et al., IEEE Trans. Evol. Comput., 2015 (GISMOE)

Le Goues et al., IEEE Trans. Software Eng., 2012 (GenProg)

Edit Sequences are Great!

Very flexible

- Easy to generate
- Easy to mutate
- Easy to crossover

Sparse

- Not source code
- Can be broken down
- *Close* to human understanding

But

- Focus on practical modification
- Mechanical

Edit Sequences are Great?

Very flexible

- Easy to generate
- Easy to mutate
- Easy to crossover

Sparse

- Not source code
- Can be broken down
- *Close* to human understanding

But

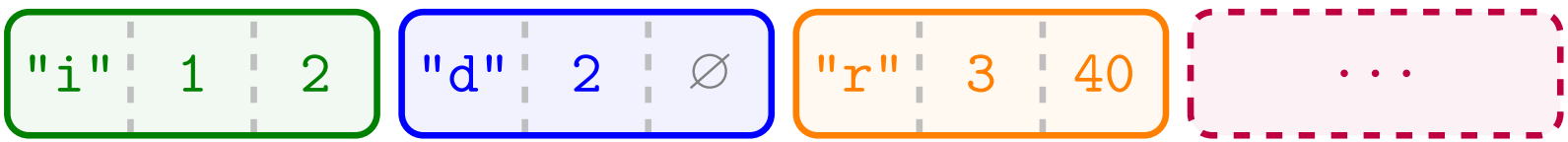
- Focus on practical modification
- Mechanical

Edit Sequences in Practice

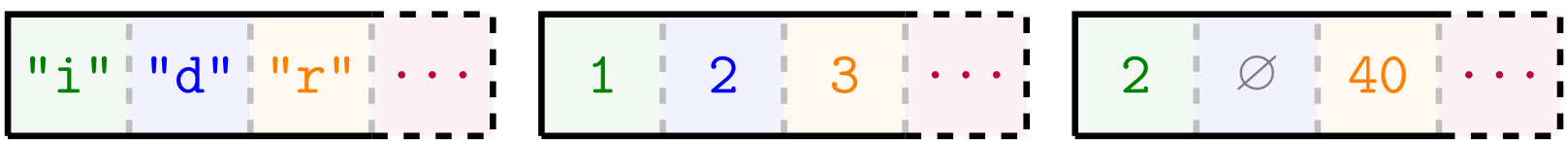
Edits as triplets `(op, t, i)`

- `op`: mutation
- `t`: target location
- `i`: ingredient location (optional)

List of triplets

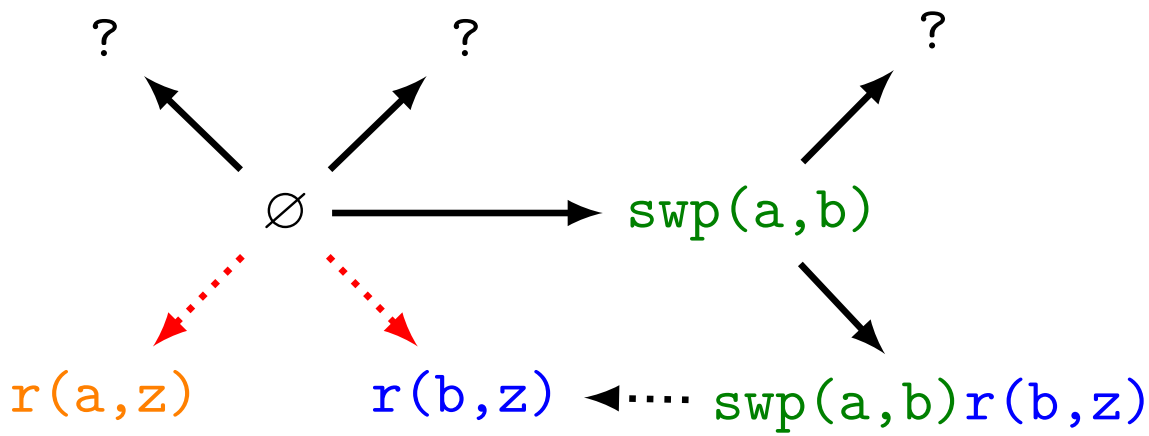


Triplet of lists



Edit Context

```
    ...
a:  x++;
b:  f(x);
    ...
z:  x = 7;
    ...
```




```
    ...
a:  f(x);
b:  x++;
    ...
z:  x = 7;
    ...
```

```
    ...
a:  f(x);
b:  x = 7;
    ...
z:  x = 7;
    ...
```

```
    ...
a:  x++;
b:  x = 7;
    ...
z:  x = 7;
    ...
```

```
    ...
a:  x = 7;
b:  f(x);
    ...
z:  x = 7;
    ...
```

Location vs Content

Motivation

Could we use content instead of/in addition to location?

- ✖ Instead of? No. Content is not unique.
- ✔ In addition to? Yes?

Examples

- Location:
  - "delete 'line 17'"
  - "replace 'line 12' by 'line 42'"
- Content:
  - "delete '`i++;`'"
  - "replace '`x = 0;`' by 'x = y;'"
- Both:
  - "delete '`i++;`' at 'line 17'"
  - "replace '`x = 0;`' at 'line 12' by 'x = y;' at 'line 42'"

Edit Semantics

Problems

- Edits are only tied to location
- Context depends on previous edits
- Context changes are ignored

Definition Proposition

Edit semantic:

Meaning in terms of content+location

`r(10,40)` “replace ‘`x++`’ at ‘line 10’ with ‘`x = 7`’ at ‘line 40’”
$\neq$ `r(10,40)` “replace ‘`f(x)`’ at ‘line 10’ with ‘`x = 7`’ at ‘line 40’”

Edit Context Changes

Patches

- patch 1: `edit1edit2edit3`
- patch 2: `edit4edit5`

During mutation

- Deletion (middle): patch 1 $\rightarrow$ `edit2edit3`
- Insertion (middle): patch 1 $\rightarrow$ `edit1edit2edit6edit3`
- Permutation: patch 1 $\rightarrow$ `edit1edit3edit2`
- Any target but the last edit

During crossover

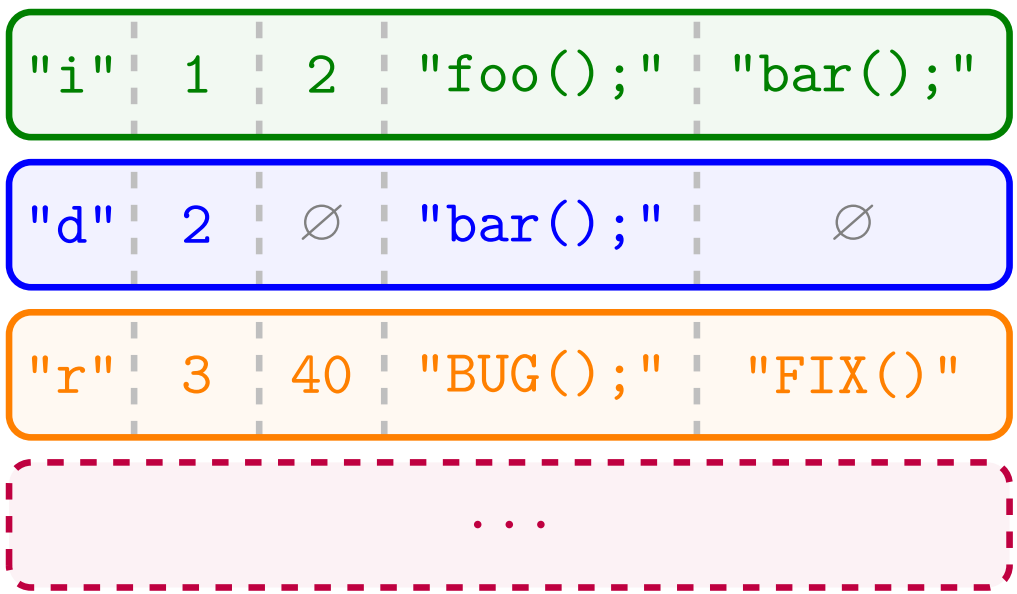
- Concatenation: `edit1edit2edit3edit4edit5`
- Any use of external data

Edit Sequences with Content

Edits as tuples `(op, tl, il, tc, ic)`

- `op`: mutation
- `tl`, `tc`: target location and content
- `il`, `ic`: ingredient location and content (optional)

List of tuples



Conflict Management

At creation

(op, a, b) becomes (op, a, b, α, β)

$\alpha = f(\mathsf{a})$, $\beta = f(\mathsf{b})$, f: lookup function

At application

Compare again α to $f(\mathsf{a})$, β to $f(\mathsf{b})$

⚠ Conflict! e.g., $\alpha \neq f(\mathsf{a})$ or $f(\mathsf{a})$ fails

- Location match: $(\mathsf{a}, \alpha) \rightarrow (\mathsf{a}, \alpha')$
- Content match: $(\mathsf{a}, \alpha) \rightarrow (\mathsf{a}', \alpha)$
- Weak match: $(\mathsf{a}, \alpha) \rightarrow (\mathsf{a}', \alpha')$

Conflict Resolution Strategies

Example: Concatenation crossover

- `edit1edit2` + `edit3edit4`
- ⚠ Conflict between edits 1 and 3! (e.g., identical)

Resolution strategies

Ignore: `edit1edit2edit3edit4`

Discard: `edit1edit2edit4`

Repair: `edit1edit2edit3'edit4`

Exploration vs exploitation

- Strategies are not mutually exclusives
- Multiple repairs can be found (and ranked)

Fuzzy Matching

For content

- String edit distance
- Tree edit distance
- Tokenization? (lexical analysis)

For location

- Euclidean distance
- Path length
- Height to target ancestor
- Specific nodes in path

Possible Further Steps?

Content repair?

- (op, a, b, α, β) $\rightarrow$ (op, a, b, α', β) $\rightarrow$ (op, a, b, α', β')
  with $\beta' = \text{repair}(\beta, \alpha, \alpha')$)

Approximate location?

- ("replace every between", a$\rightarrow$a', b, α, β)
- ("replace every inside", a, b, α, β)

Partial edits?

- ("replace", a, b, "□ < □", "□ <= □")
- ("delete", □, $\varnothing$, "assert(□)", $\varnothing$)

Final Words

Edit sequences are great

- Very flexible
- Sparse
- Too mechanical?

Content based semantic

- More guidance
- More diversity
- Yet to be investigated

Selected References

William B. Langdon and Mark Harman.

Optimizing existing software with genetic programming.

IEEE Transactions on Evolutionary Computation, 19(1):118–135, 2015.

Claire Le Goues, ThanhVu Nguyen, Stephanie Forrest, and Westley Weimer.

GenProg: A generic method for automatic software repair.

IEEE Transactions on Software Engineering, 38(1):54–72, 2012.

Vinicius Paulo L. Oliveira, Eduardo Faria de Souza, Claire Le Goues, and Celso G. Camilo-Junior.

Improved representation and genetic operators for linear genetic programming for automated program repair.

Empirical Software Engineering, 23(5):2980–3006, 2018.

Justyna Petke, Saemundur O. Haraldsson, Mark Harman, William B. Langdon, David Robert White, and John R. Woodward.

Genetic improvement of software: A comprehensive survey.

IEEE Transactions on Evolutionary Computation, 22(3):415–432, 2018.