Clone detection

Designing and Maintaining Software (DAMS)

Louis Rose

Habitable Software

Leaner

Less **Complex**

Loosely **Coupled**

More **Cohesive**

Avoids **Duplication**

Clearer

More **Extensible**

???

Goal

Automatically detect similar fragments of code.

```
class StuffedCrust
  def title
    "Stuffed Crust " +
      @toppings.title +
      " Pizza"
  end

  def cost
    @toppings.cost + 6
  end
end
```

```
class DeepPan
  def title
    "Deep Pan " +
      @ingredients.title +
      " Pizza"
  end

  def cost
    @ingredients.cost + 4
  end
end
```

Goal

Automatically detect similar fragments of code.

```
class StuffedCrust
  def title
    "Stuffed Crust " +
      @toppings.title +
      " Pizza"
  end

  def cost
    @toppings.cost + 6
  end
end
```

```
class DeepPan
  def title
    "Deep Pan " +
      @ingredients.title +
      " Pizza"
  end

  def cost
    @ingredients.cost + 4
  end
end
```

Beginnings

Automatically detect **identical** fragments of **text**.

```
class StuffedCrust
  def title
    "Stuffed Crust " +
      @toppings.title +
      " Pizza"
  end

  def cost
    @toppings.cost + 6
  end
end
```

```
class DeepPan
  def title
    "Deep Pan " +
      @ingredients.title +
      " Pizza"
  end

  def cost
    @ingredients.cost + 4
  end
end
```

Beginnings

Automatically detect **identical** fragments of **text**.

```
class StuffedCrust
  def title
    @toppings.title +
    " Pizza"
  end

  def cost
    @toppings.cost + 6
  end
end
```

```
class DeepPan
  def title
    @toppings.title +
    " Pizza"
  end

  def cost
    @toppings.cost + 4
  end
end
```

Beginnings

Automatically detect **identical** fragments of **text**.

```
class StuffedCrust
  def title
    @toppings.title +
      " Pizza"
  end

  def cost
    @toppings.cost + 6
  end
end
```

```
class DeepPan
  def title
    @toppings.title +
      " Pizza"
  end

  def cost
    @toppings.cost + 4
  end
end
```

Comparing Fragments

Simple approach: identify lines of A appearing in B.

```
class StuffedCrust
  def title
    @toppings.title +
    " Pizza"
  end

  def cost
    @toppings.cost + 6
  end
end
```

```
class DeepPan
  def title
    @toppings.title +
    " Pizza"
  end

  def cost
    @toppings.cost + 4
  end
end
```

Comparing Fragments

Simple approach: identify lines of A appearing in B.

```
class StuffedCrust
  def title
    @toppings.title +
    " Pizza"
  end

  def cost
    @toppings.cost + 6
  end
end
```

```
class DeepPan
  def title
    @toppings.title +
    " Pizza"
  end

  def cost
    @toppings.cost + 4
  end
end
```

Comparing Fragments

Simple approach: identify lines of A appearing in B.

```
class StuffedCrust
  def title
    @toppings.title +
    " Pizza"
  end

  def cost
    @toppings.cost + 6
  end
end
```

```
class DeepPan
  def title
    @toppings.title +
    " Pizza"
  end

  def cost
    @toppings.cost + 4
  end
end
```

Comparing Fragments

Simple approach: identify lines of A appearing in B.

```
class StuffedCrust
  def title
    @toppings.title +
    " Pizza"
  end

  def cost
    @toppings.cost + 6
  end
end
```

```
class DeepPan
  def title
    @toppings.title +
    " Pizza"
  end

  def cost
    @toppings.cost + 4
  end
end
```

Comparing Fragments

Simple approach: identify lines of A appearing in B.

```
class StuffedCrust
  def title
    @toppings.title +
    " Pizza"
  end

  def cost
    @toppings.cost + 6
  end
end
```

```
class DeepPan
  def title
    @toppings.title +
    " Pizza"
  end

  def cost
    @toppings.cost + 4
  end
end
```

Comparing Fragments

Simple approach: identify lines of A appearing in B.

```
class StuffedCrust
  def title
    @toppings.title +
    " Pizza"
  end

  def cost
    @toppings.cost + 6
  end
end
```

```
class DeepPan
  def title
    @toppings.title +
    " Pizza"
  end

  def cost
    @toppings.cost + 4
  end
end
```

Comparing Fragments

Simple approach: identify lines of A appearing in B.

```
class StuffedCrust
 def title
    @toppings.title +
    " Pizza"
 end

 def cost
   @toppings.cost + 6
 end
end
```

```
class DeepPan
 def title
    @toppings.title +
    " Pizza"
 end

 def cost
   @toppings.cost + 4
 end
end
```

Comparing Fragments

Increase fragment size until no new clones appear.

```
class StuffedCrust
  def title
    @toppings.title +
    " Pizza"
  end

  def cost
    @toppings.cost + 6
  end
end
```

```
class DeepPan
  def title
    @toppings.title +
    " Pizza"
  end

  def cost
    @toppings.cost + 4
  end
end
```

Results

The simple approach indicates the following clones:

```
class StuffedCrust
  def title
    @toppings.title +
      " Pizza"
  end

  def cost
    @toppings.cost + 6
  end
end
```

```
class DeepPan
  def title
    @toppings.title +
      " Pizza"
  end

  def cost
    @toppings.cost + 4
  end
end
```

Tweaking

From now on, strip out whitespace and comments

```
class StuffedCrust
  def title
    @toppings.title +
    " Pizza"
  end
  def cost
    @toppings.cost + 6
  end
end
```

```
class DeepPan
  def title
    @toppings.title +
    " Pizza"
  end
  def cost
    @toppings.cost + 4
  end
end
```

More Tweaking

Ignore fragments containing only Ruby keywords.

```
class StuffedCrust
  def title
    @toppings.title +
    " Pizza"
  end
  def cost
    @toppings.cost + 6
  end
end
```

```
class DeepPan
  def title
    @toppings.title +
    " Pizza"
  end
  def cost
    @toppings.cost + 4
  end
end
```

Recap: Goal

Automatically detect similar fragments of code.

```
class StuffedCrust
  def title
    "Stuffed Crust " +
      @toppings.title +
      " Pizza"
  end

  def cost
    @toppings.cost + 6
  end
end
```

```
class DeepPan
  def title
    "Deep Pan " +
      @ingredients.title +
      " Pizza"
  end

  def cost
    @ingredients.cost + 4
  end
end
```

Parameterisation

Alter the source before clone detection

```
class StuffedCrust
  def title
    "Stuffed Crust " +
      @toppings.title +
      " Pizza"
  end

  def cost
    @toppings.cost + 6
  end
end
```

```
class DeepPan
  def title
    "Deep Pan " +
      @ingredients.title +
      " Pizza"
  end

  def cost
    @ingredients.cost + 4
  end
end
```

Parameterisation

Alter the source before clone detection

```
class C
  def title
    "Stuffed Crust " +
      @toppings.title +
      " Pizza"
  end

  def cost
    @toppings.cost + 6
  end
end
```

```
class C
  def title
    "Deep Pan " +
      @ingredients.title +
      " Pizza"
  end

  def cost
    @ingredients.cost + 4
  end
end
```

Parameterisation

Alter the source before clone detection

```
class C
  def M
    "Stuffed Crust " +
      @toppings.title +
      " Pizza"
  end

  def cost
    @toppings.cost + 6
  end
end
```

```
class C
  def M
    "Deep Pan " +
      @ingredients.title +
      " Pizza"
  end

  def cost
    @ingredients.cost + 4
  end
end
```

Parameterisation

Alter the source before clone detection

```
class C
  def M
    S +
      @toppings.title +
      " Pizza"
  end

  def cost
    @toppings.cost + 6
  end
end
```

```
class C
  def M
    S +
      @ingredients.title +
      " Pizza"
  end

  def cost
    @ingredients.cost + 4
  end
end
```

Parameterisation

Alter the source before clone detection

```
class C
  def M
    S +
      @A.title +
      " Pizza"
  end

  def cost
    @toppings.cost + 6
  end
end
```

```
class C
  def M
    S +
      @A.title +
      " Pizza"
  end

  def cost
    @ingredients.cost + 4
  end
end
```

Parameterisation

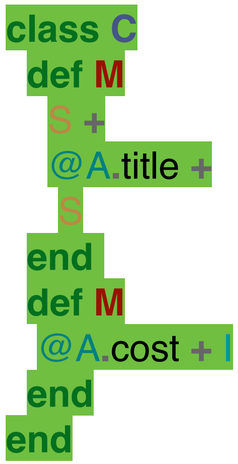
Alter the source before clone detection

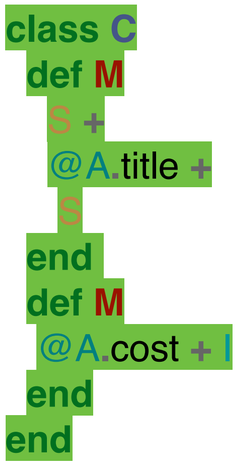
```
class C
  def M
    S +
      @A.title +
      S
  end

  def M
    @A.cost + I
  end
end
```

```
class C
  def M
    S +
      @A.title +
      S
  end

  def M
    @A.cost + I
  end
end
```

Clone Detection

Now perform detection on parameterised source.



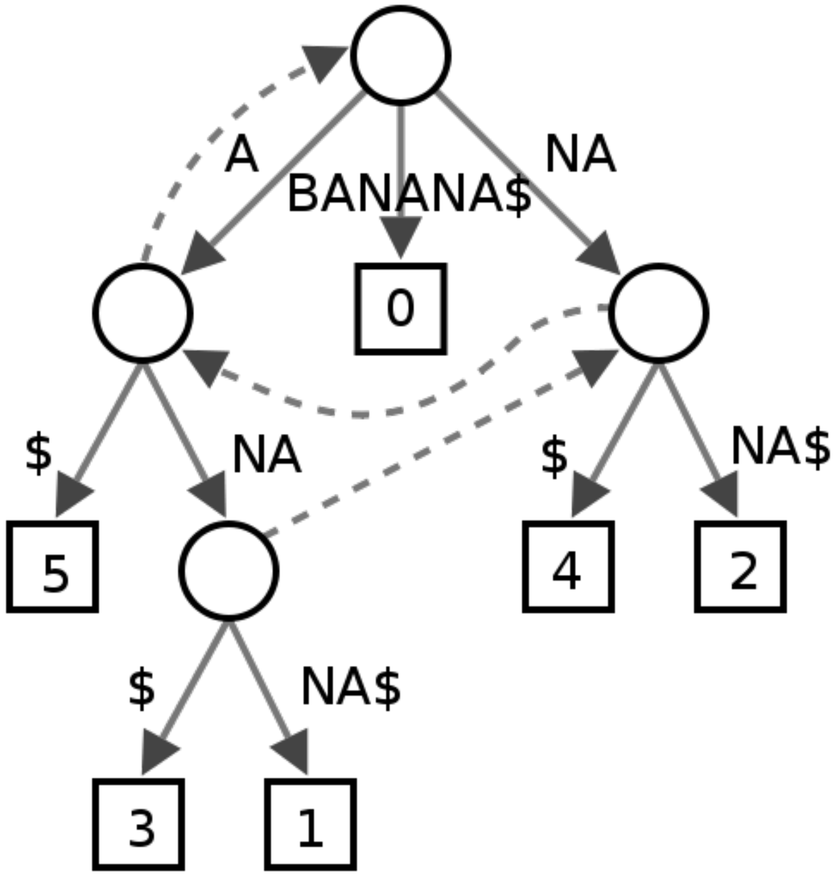


Baker's Algorithm

"Duplication may be introduced into a large system as modifications are made to add new features or to fix bugs. Rather than rewrite working sections of code, programmers may copy and modify sections of code. It has long been known that copying sections of code may make the code larger, more complex, and more difficult to maintain."

- Brenda S. Baker
A Program for Identifying Duplicated Code
Computing Science and Statistics, 24, 1992

Performance



https://en.wikipedia.org/wiki/Suffix_tree

Limitations

Baker's algorithm cannot detect syntactically distinct but semantically equivalent clones

```
class StuffedCrust
  def title
    "Stuffed Crust " +
      @toppings.title +
      " Pizza"
  end

  def cost
    @toppings.cost + 6
  end
end
```

```
class Stonebaked
  def cost
    6 + @toppings.cost
  end

  def title
    "Stonebaked #{@toppings.title} Pizza"
  end
end
```

Clones are Indicators

No algorithm can definitively determine whether a clone is accidental or essential duplication

```
class StuffedCrust
  def title
    "Stuffed Crust " +
      @toppings.title +
      " Pizza"
  end

  def cost
    @toppings.cost + 6
  end
end
```

```
class Stonebaked
  def cost
    6 + @toppings.cost
  end

  def title
    "Stonebaked #{@toppings.title} Pizza"
  end
end
```

Clones are Indicators

No algorithm can definitively determine whether a clone is accidental or essential duplication

```
class StuffedCrust
  def title
    "Stuffed Crust " +
      @toppings.title +
      " Pizza"
  end

  def cost
    @toppings.cost + 6
  end
end
```

```
class Stonebaked
  def cost
    6 + @toppings.cost
  end

  def title
    "Stonebaked #{@toppings.title} Pizza"
  end
end
```

Clones are Indicators

No algorithm can definitively determine whether a clone is accidental or essential duplication

```
class StuffedCrust
  def title
    "Stuffed Crust " +
      @toppings.title +
      " Pizza"
  end

  def cost
    @toppings.cost + 6
  end
end
```

```
class Stonebaked
  def cost
    6 + @toppings.cost
  end

  def title
    "Stonebaked #{@toppings.title} Pizza"
  end
end
```

Summary

Clones can indicate potentially duplicated code

Baker's is the most straightforward clone detection algorithm, and there are many more sophisticated

No tool can distinguish accidental and essential duplication