Tree differencing for code copy detection

Nicolas Manichon
nicolas.manichon@lse.epita.fr

Overview

- The project
- The challenges
- Existing tools
- How it works

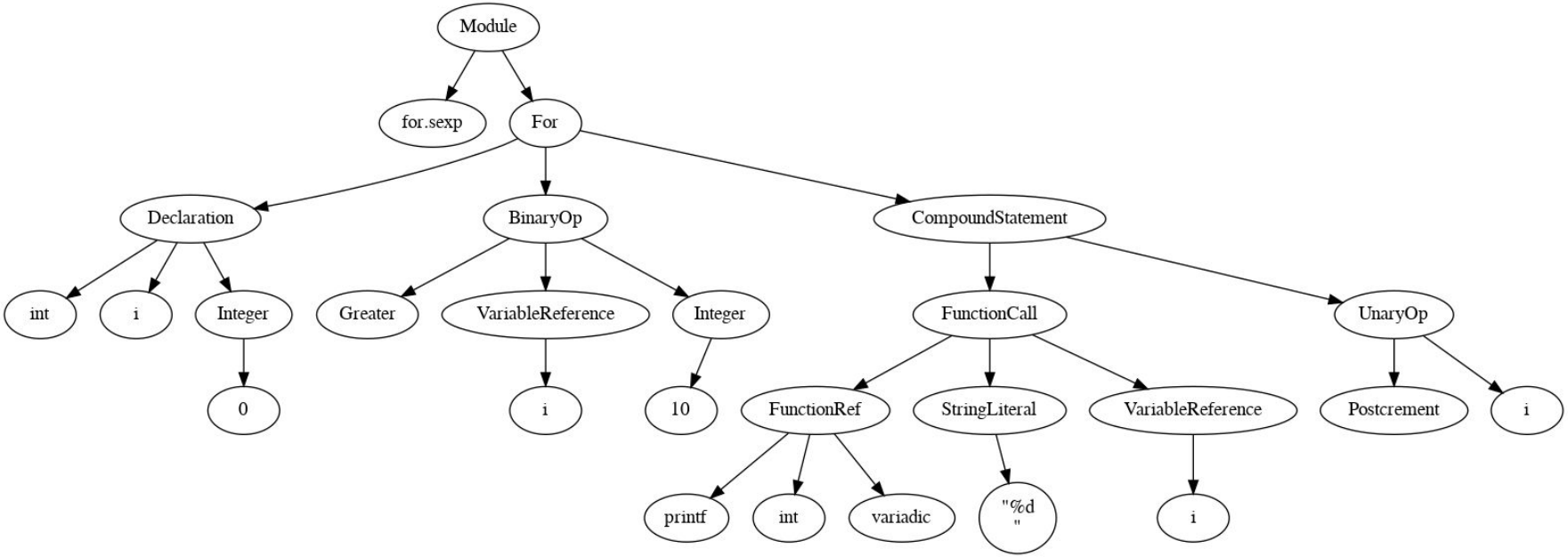
The project

Code copy detection can be used in security and cheating detection.

I wanted to learn about Clang/LLVM.

The challenges: Representation

I want to work at the AST level (and I want a generic AST!)



The challenges: Scaling

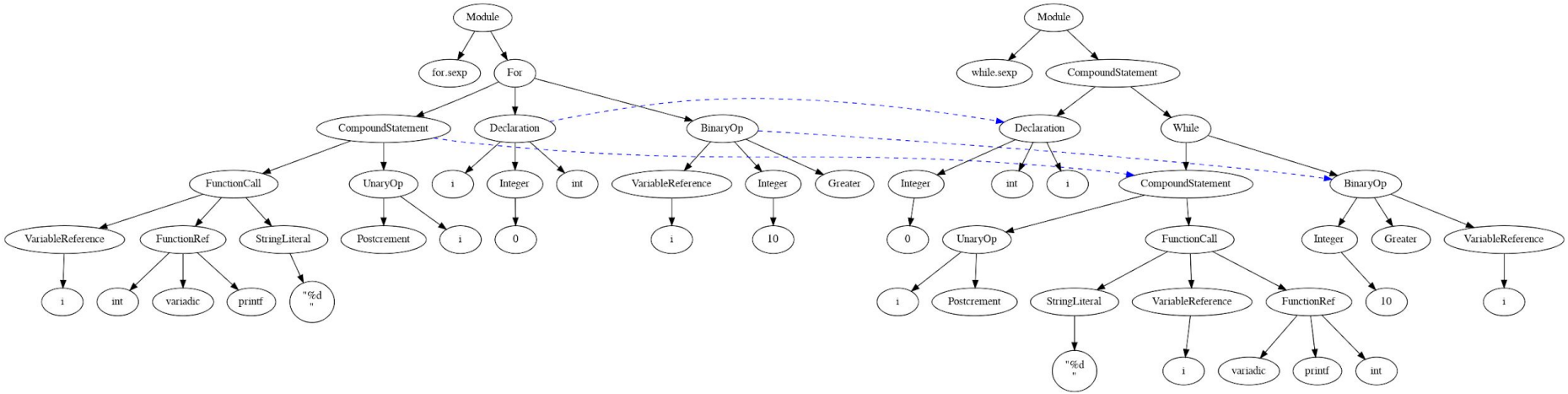
Cheating detection scales very fast: 400 submissions of 10 files each.

$$\binom{2}{400} * (10^2) = 7\,980\,000$$

The challenges: Robustness

Cheaters aren’t very smart.

But they can change variable names.



The challenges: Output

I want to produce usable output.

I need location info, file names...

JSON output, and a web interface.


```
[
    {
        "directory1": "krendus/k-1",
        "directory2": "krendus/k-10",
        "matches": [
            {
                "file1": {
                    "path": "/home/nicolas/Documents/Prog/ast-diff/krendus/k-1/k/k.c.sexp",
                    "directory": "krendus/k-1"
                },
                "file2": {
                    "path": "/home/nicolas/Documents/Prog/ast-diff/krendus/k-10/k/k.c.sexp",
                    "directory": "krendus/k-10"
                },
                "similarity": "0.70922",
                "locations": [
                    {
                        "file1loc": "k.c:33:1 k.c:49:1",
                        "file2loc": "k.c:33:1 k.c:51:1"
                    }
                ]
            }
        ]
    },
    {
        "directory1": "krendus/k-1",
        "directory2": "krendus/k-11",
        "matches": [
            {
                "file1": {
                    "path": "/home/nicolas/Documents/Prog/ast-diff/krendus/k-1/k/k.c.sexp",
                    "directory": "krendus/k-1"
                },
                "file2": {
                    "path": "/home/nicolas/Documents/Prog/ast-diff/krendus/k-11/k/k.c.sexp",
                    "directory": "krendus/k-11"
                },
                "similarity": "0.719424",
                "locations": [
                    {
                        "file1loc": "k.c:33:1 k.c:49:1",
                        "file2loc": "k.c:32:1 k.c:53:1"
                    }
                ]
            }
        ]
    },
```

Existing tools: jPlag

Specialized in plagiarism detection.

Implements an ANTLR parser for every language it supports.

Supports Java, C#, C, C++, Scheme.

Web interface.

Matches for 862564 & 862326

35.9%

INDEX - HELP

Jumpbox.java(99-120)	Jumpbox.java(307-332)	17
Jumpbox.java(161-177)	Jumpbox.java(181-194)	13
Jumpbox.java(188-205)	Jumpbox.java(207-224)	16
Jumpbox.java(229-246)	Jumpbox.java(238-255)	12
Jumpbox.java(247-269)	Jumpbox.java(139-162)	31

```
public void paint( Graphics g)
{
  super.paint( g );

  // zeichne den weissen Rahmen
  g.setColor( Color.white);
  g.drawRect( mInsets.left, mInsets.top + 50, 500, 450);

  // Zeichne den Jumper
  g.setColor( mColors[mRichtungsFarbe[ mRichtung ]]);
  g.fillRect( mXPos + mInsets.left, mYPos + mInsets.top + 50, 50, 50);

  // Zeichne die 4 Quadrate
  for( int i = 0; i < 4 ; i++)
  {
    g.setColor( mColors[mRichtungsFarbe[i]]);
    g.fillRect( i * 50 + mInsets.left, mInsets.top, 50, 50);
    g.setColor( Color.black);
    g.drawString( mRichtungsName[i], i * 50 + 20 + mInsets.left, 30 + mInsets.
  }
}

public boolean handleEvent( Event inEvt )
{
  boolean result = false;
  switch ( inEvt.id )
  {
      case Event.MOUSE_MOVE:
      case Event.MOUSE_DRAG:
      case Event.MOUSE_DOWN:
      case Event.MOUSE_UP:
      case Event.MOUSE_ENTER:
      case Event.MOUSE_EXIT:
        mMouseX = inEvt.x - mInsets.left;
        mMouseY = inEvt.y - mInsets.top - 50;
        handleMouse( mMouseX, mMouseY );
        result = true;
```



```
  public void paint(Graphics g)
  {
    super.paint(g);
    // Hintergrund zeichnen:
    // Grundfarbe
    g.setColor(hintergrundFarbe);
    g.fillRect(jbInsets.left ,jbInsets.top,
               500 + jbInsets.left ,400 + jbInsets.top);
    // Rahmen
    g.setColor(Color.darkGray);
    g.drawRect(jbInsets.left, 50 + jbInsets.top,
               499 + jbInsets.left ,349 + jbInsets.top);

    // die vier Quadrate zeichnen:
    for (int i=0; i<4; i++)
    {
      // die Farbquadrate
      g.setColor(farben[fIndex[i]]);
      g.fillRect(jbInsets.left + i*50, jbInsets.top, 50, 50);
      // ... und die Beschriftung
      g.setColor(Color.black); // Schwarz fuer guten Kontrast
      g.drawString(riName[i], jbInsets.left + i*50 + 22, jbInsets.top + 32);
    }

    // DIE Box zeichnen
    g.setColor(farben[fIndex[richtung]]);
    g.fillRect(boxX + jbInsets.left, boxY + jbInsets.top, 50, 50);
  }
}
```

Existing tools: clang-diff

LibTooling based program.

Only supports the languages supported by Clang (C, C++, ObjC).

Only works with 2 files at a time.

No consumable output, but a simple web interface.

```
struct abc {
        int a;
};

struct pasdefini;

void fonctioninconnu(struct pasdefini* p, int z, struct abc* a);

int test(void);

union z {
        int a;
        int b;
        float c;
};

#define LOL (20)

int main(void) {
        int a;
        a = 20;
        return test();
}
```


```
struct abc {
        int a;
};

struct pasdefini;

void fonctioninconnu(struct pasdefini* p, int z, struct abc* a);

int test(void);

union z {
        int a;
        int b;
        float c;
};

#define LOL (20)

int main(void) {
        int a;
        a = 20;
        return test();
}
```

How it works: The gumtree algorithm

“Fine-grained and Accurate Source Code Differencing” by Jean-Rémy Falleri, etc.

Used by jPlag, clang-diff, and me!

Pseudo code in the paper, and a reference implementation on Github.

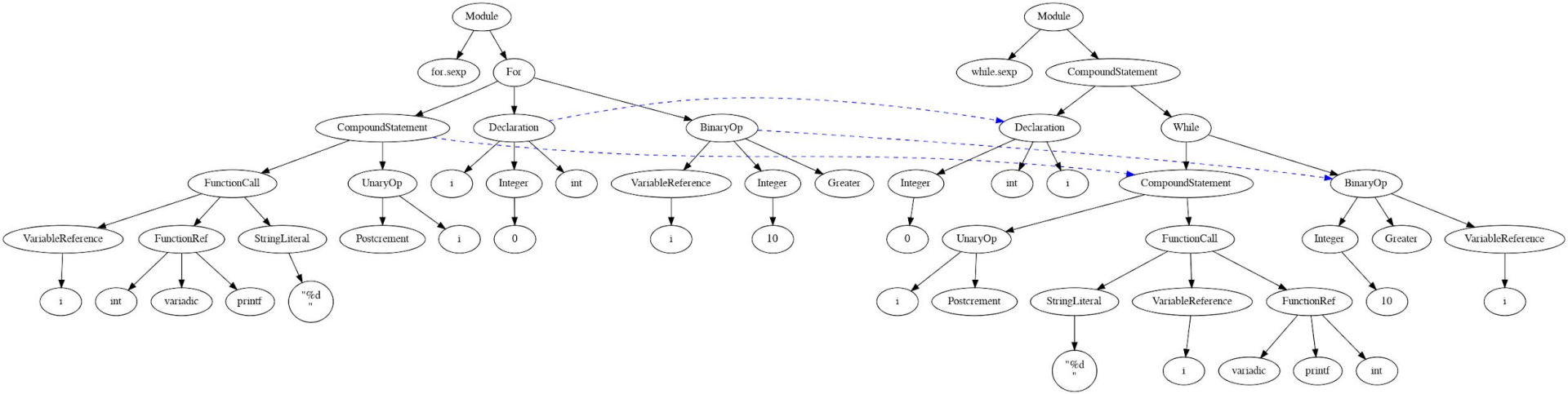
How it works: The gumtree algorithm

Two phases, top down and bottom up.

We first go down the tree, matching greedily.

And we then go up the tree, matching parents of matched nodes.

$O(N^2)$


Module
for.sexp
For
CompoundStatement
Declaration
BinaryOp
FunctionCall
UnaryOp
i
Integer
int
VariableReference
Integer
Greater
VariableReference
FunctionRef
StringLiteral
Postcrement
i
0
i
10
i
int
variadic
printf
"%d
"
Module
while.sexp
CompoundStatement
Declaration
While
Integer
int
i
CompoundStatement
BinaryOp
0
UnaryOp
FunctionCall
Integer
Greater
VariableReference
i
Postcrement
StringLiteral
VariableReference
FunctionRef
10
i
"%d
"
i
variadic
printf
int

How it (doesn't) work: The gumtree algorithm

$$\text{dice}(t_1, t_2, \mathcal{M}) = \frac{2 \times |\{t_1 \in s(t_1) | (t_1, t_2) \in \mathcal{M}\}}{|s(t_1)| + |s(t_2)|}$$

if isomorphic(t_1, t_2) **then**
  if $\exists t_x \in T_2 \mid \text{isomorphic}(t_1, t_x) \wedge t_x \neq t_2$
  or $\exists t_x \in T_1 \mid \text{isomorphic}(t_x, t_2) \wedge t_x \neq t_1$
  then
    $add(A, (t_1, t_2))$;
  else
    add all pairs of isomorphic nodes of $s(t_1)$
    and $s(t_2)$ to $\mathcal{M}$;

How it works: clang-sexpression

A LibTooling based program.

Runs as a syntax action over the files passed as parameters, and outputs s-expressions and location info.

S-expressions are a generic way to represent trees, and are easy to parse.

```
(Salut (je suis) "Nicolas?")
```

I wrote my own “visitors”, that gave me more control over the traversal.

```c
#include <stdio.h>

int main(int argc, char *argv[])
{
        for (int i = 0; i < 10; i++)
                printf("%d\n", i);
}
```

```
      for.c:begin for.c:end
      for.c:begin for.c:end
      for.c:4:1 for.c:7:1
      for.c:4:1 for.c:7:1
      for.c:3:1 for.c:7:1
      for.c:3:1 for.c:7:1
      for.c:3:10 for.c:3:14
      for.c:3:10 for.c:3:14
      for.c:3:10 for.c:3:14
      for.c:3:20 for.c:3:31
      for.c:3:20 for.c:3:31
      for.c:3:20 for.c:3:31
      for.c:4:1 for.c:7:1
      ...
```

```
(TranslationUnit
  ("for.c")
  (Function
   (main)
   ("int (int, char **)")
   (FunctionParameters
    (ParmVarDecl
     (argc)
     ("int"))
    (ParmVarDecl
     (argv)
     ("char **")))
   (CompoundStmt
    (ForStmt
     (DeclStmt
      (VarDecl
       (i)
       (IntegerLiteral
        (0)
        ("int"))

      ...
```

How it works: ast-diff

ast-diff is a tool that reads s-expressions from files, and compares them.

To compare C and C++ specifically, I wrote a tool that encapsulates clang-sexpression and ast-diff. It uses as many CPU cores as possible.

I wrote another python script that can display the results in a web interface.

```
42sh$ ast-diff --diff test/for.sexp test/while.sexp \
            test/while.sexp test/for.sexp
```

```
{
    "results": [
        {
            "file1": "test/for.sexp",
            "file2": "test/while.sexp",
            "similarity": 0.872727,
            "mappings": [
                ...
            ]
        },
        {
            "file1": "test/while.sexp",
            "file2": "test/for.sexp",
            "similarity": 0.872727,
            "mappings": [
                ...
            ]
        }
    ]
}
```

```
42sh$ dispatch krendus/* --jobs=42 --glob='*.c' \
      --ex-glob='givenfile.c'
```

```
[
    {
        "directory1": "krendus/k-1",
        "directory2": "krendus/k-12",
        "matches": []
    },
    {
        "directory1": "krendus/k-1",
        "directory2": "krendus/k-13",
        "matches": [
            {
                "file1": {
                    "path": "krendus/k-1/k/io.c.sexp",
                    "directory": "krendus/k-1"
                },
                "file2": {
                    "path": "krendus/k-13/k/serial.c.sexp",
                    "directory": "krendus/k-13"
                },
                "similarity": "0.693671",
                "locations": [
                    {
                        "file1loc": "io.c:10:8 io.c:20:1",
                        "file2loc": "serial.c:15:4 serial.c:25:1"
                    }
                ]
            }
        ...
```

Demo

Conclusion

I learned about LibTooling, and got a patch merged into Clang.

I had to solve a few interesting challenges.

What’s left to do:

- More options for clang-sexpression.
- A few performance optimizations.
- Handling code that does not compile.
- Support more languages.
- Learn CSS.

Questions?

ast-diff: https://github.com/balayette/ast-diff

clang-sexpression: https://github.com/balayette/clang-sexpression

Fine-grained and Accurate Source Code Differencing: https://hal.archives-ouvertes.fr/hal-01054552/document

GumtreeDiff: https://github.com/GumTreeDiff/gumtree

LibTooling: https://clang.llvm.org/docs/LibTooling.html