Binary clone detection

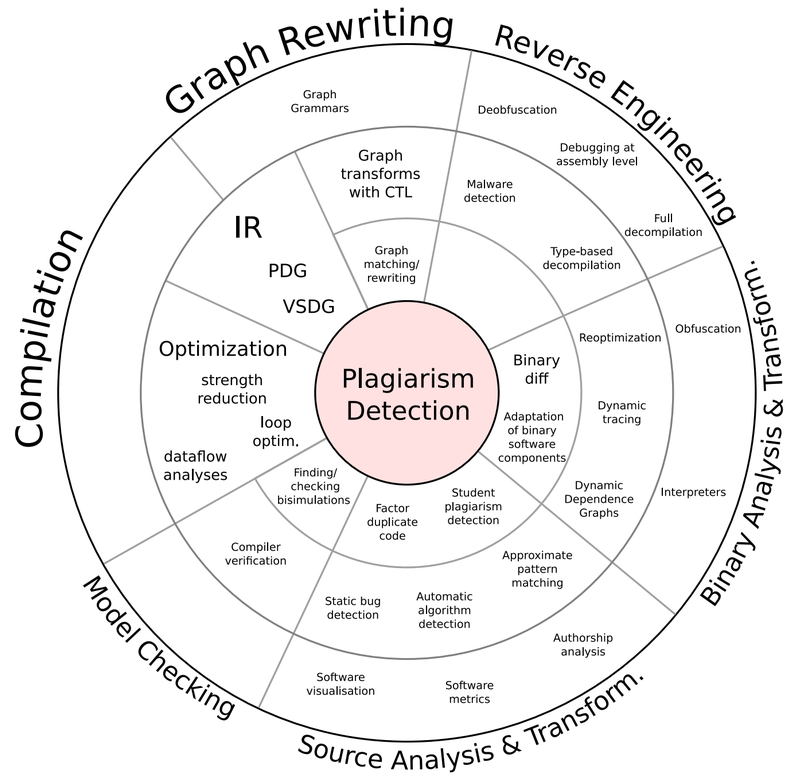
Christopher Gautier
under the supervision of Prof. Alan Mycroft
Computer Laboratory, University of Cambridge

Software plagiarism

- Open source software is particularly vulnerable to code-theft. Unscrupulous companies might be tempted to reduce development costs by taking existing open-source code and incorporating it into commercial opaque binaries without credits or authorization.
- If source code is not available, code from executables and libraries can still be extracted and incorporated into another project, using binary software adaptation techniques.

Whereas source code plagiarism has been relatively well studied, **binary clone detection remains effectively an unsolved problem**, notably due to the greater complexity of analyzing optimized machine code or bytecode.

Related themes

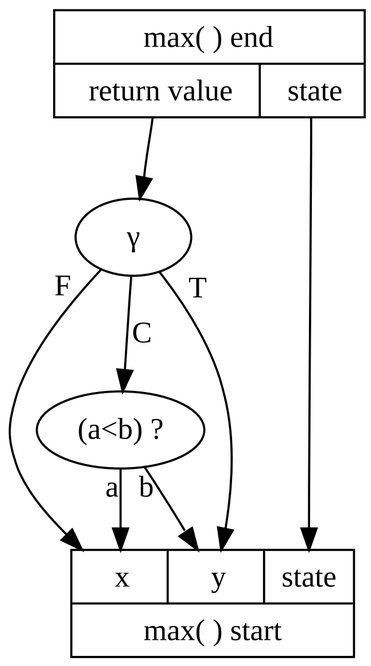


Value State Dependence Graphs

The VSDG is basically the VDG IR of Weise *et al.* (*cf.* "Value Dependence Graphs: Representation without Taxation"), augmented with explicit **state nodes and dependencies** to represent R/W to the heap, I/O instruction sequences, or any other stateful information.

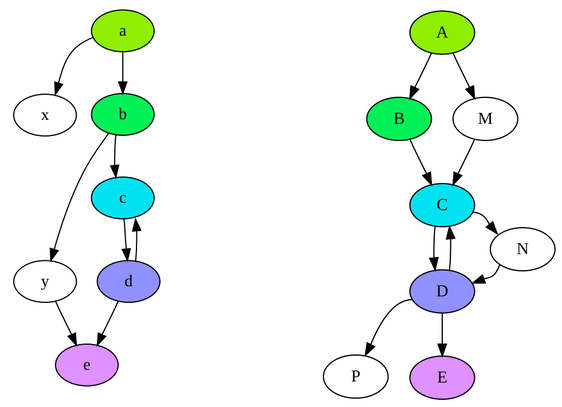
The VSDG follows the trend of making control-flow representation more implicit, and data-flow dependences more explicit, hence producing **more normalizing representations**.

Historically:
CFG → PDG → SSA → VDG → VSDG



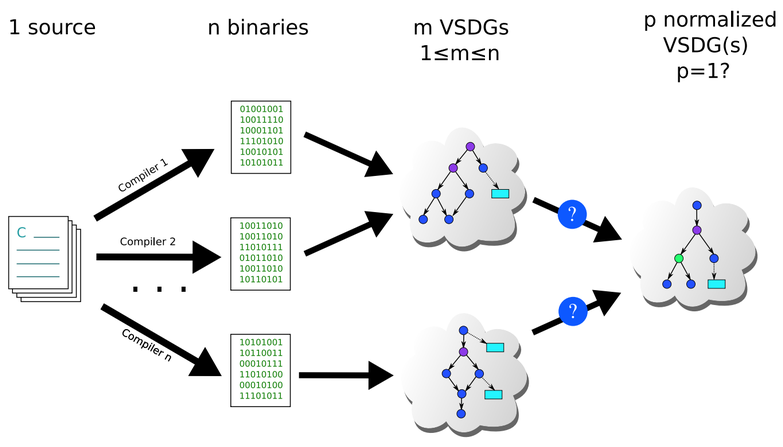
Our motivation:

- If we use a "normalizing-enough" IR, two different programs with identical semantics may have the same representation.
- The clone detection task becomes a **subgraph isomorphism problem**.



Current status

Certain classes of optimization (*e.g.* basic block reordering, loop transformations *etc.*) introduce variability in the VSDGs.



Therefore **VSDGs must be further normalized**. Can we do undo compiler passes with graph rewriting rules, or has the compilation process thrown away too much information?

Case study: loop inversion

```
I;                          I;
while(cond){U;}     ⇒       if(cond') do{U;}while(cond);
```