Applying formal methods to analysis of semantic differences between versions of software

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Ing. František Nečas
Supervisor: Ing. Viktor Malík, Ph.D.

DiffKemp

DiffKemp: Static Analysis of Semantic Differences of Large-scale C Projects

- Some projects must maintain **semantic stability** between versions, for example:
  - System libraries (e.g. the standard C library) whose users rely on their functionality.
  - Functions in the RHEL kernel that are a part of the Kernel Application Binary Interface.
- We want to **automatically** check that the semantics of certain functions was not modified.
- Tools based on **formal methods** are very **precise but far too slow**.
- DiffKemp: open-source **highly scalable** framework for identifying semantic differences.

Are the following functions semantically equal?

renamed variable

inverse condition + swapped branches

```
int f(int a) {
  int r;
  r = 0;
  if (a > 100) {
    r = a - 10;
  } else {
    r = f(a + 11);
    r = f(r);
  }
  return r;
}
```

```
int f(int x) {
  int r;
  r = 0;
  if (x < 101) {
    r = f(11 + x);
    r = f(r);
  } else {
    r = x - 10;
  }
  return r;
}
```

The Basic Comparison Algorithm

The analysis in DiffKemp is built on several concepts:

- The versions are compiled into the **LLVM Intermediate Representation** (IR) to make the comparison simpler.
- Where possible, versions are compared **instruction-by-instruction**.
- DiffKemp contains a number of pre-defined **change patterns** that are known to preserve semantics (e.g., refactoring a code block into a new function).

Results and Experiments

Evaluated on simple hand-made programs, the EqBench benchmark and the Linux kernel. **EqBench results**:

	SMT Off	SMT On
Correct equal	57	62
Correct not-equal	125	125
Incorrect not-equal	90	85
Incorrect equal	0	0

Integrating Formal Methods into the Analysis Core

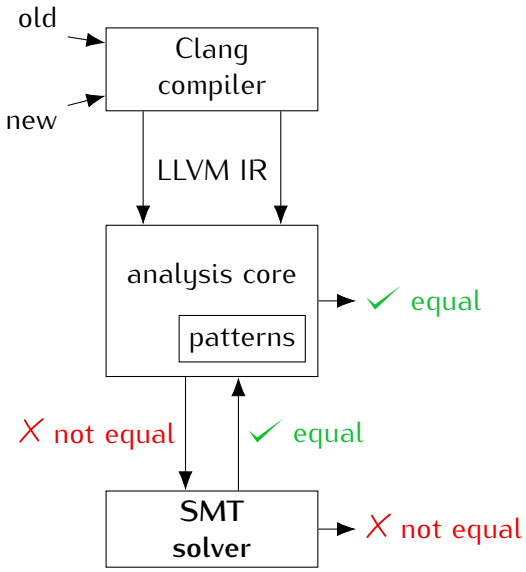
- The built-in patterns do **not cover all refactorings**.
- We aim to check equality of **complex arithmetic and logic** changes (e.g., distributive properties).
- When a difference is found and **no pattern** is available, **encode** the equivalence of the following blocks **into a formula**:

$$\bigwedge_{v_1 \in InVar_1} v_1 = varmap(v_1) \wedge \quad \text{Input equality}$$

$$Block_1 \wedge Block_2 \wedge \quad \text{Encoded blocks}$$

$$\neg \bigwedge_{out_1 \in OutVar_1} out_1 = outmap(out_1) \quad \text{Output equality}$$

- Use an **SMT solver** to check satisfiability. The blocks are equal iff the formula is unsatisfiable.



The project was supported by Red Hat Research.
frantisek.necas@protonmail.com