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

BRNO FACULTY OF INFORMATION TECHNOLOGY UNIVERSITY OF TECHNOLOGY

František Nečas
Supervisor: Ing. Viktor Malík

Excel@FIT 2024

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

- Some projects must maintain **semantic stability** between versions, e.g., system libraries.
- We want to **automatically** check that the semantics of certain functions was not modified.
- Tools based on **formal methods** are very **precise but slow**.
- DiffKemp: open-source **highly scalable** framework for identifying semantic differences.

Are the following functions semantically equal?

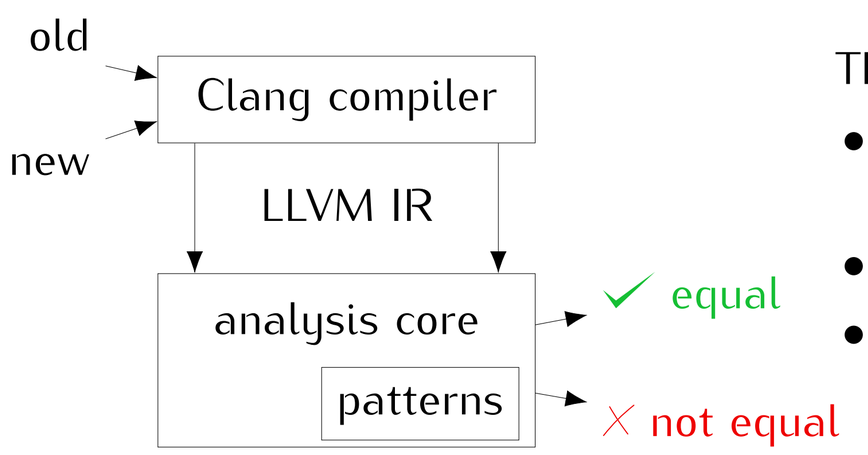
renamed variable

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:

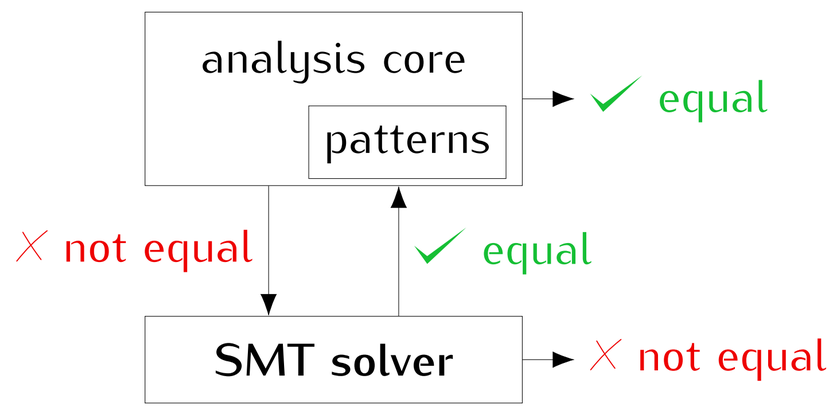
- The versions are compiled into the **LLVM Intermediate Representation** (IR) to make the comparison simpler.
- Where possible, versions are compared **instruction-by-instruction**.
- DiffKemp contains a number of pre-defined **change patterns** that are known to preserve semantics (e.g., refactoring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

Integrating Formal Methods into the Analysis Core

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

$$size_1 = size_2 \wedge offset_1 = offset_2 \wedge val_1 = val_2 \wedge$$
$$mask_1 = (((1 << size_1) - 1) << offset_1) \wedge ret_1 = ((val_1 \,\&\, mask_1) >> offset_1) \wedge$$
$$mask_2 = (1 << size_2) - 1 \wedge ret_2 = (val_2 >> offset_2) \,\&\, mask_2 \wedge$$
$$\neg(ret_1 = ret_2)$$

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



Results and Experiments

	SMT Off	SMT On
Correct equal	56	60
Correct not-equal	125	125
Incorrect not-equal	91	87
Incorrect equal	0	0

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

```
if (reg & 0xff000000) {
    unsigned char size, offset;
    size = (reg >> 24) & 0x3f;
    offset = (reg >> 16) & 0x1f;
    mask = ((1 << size) - 1) << offset;
    return (val & mask) >> offset;
} else {
    return val;
}
```

```
if (reg & 0xff000000) {
    unsigned char size, offset;
    size = (reg >> 24) & 0x3f;
    offset = (reg >> 16) & 0x1f;
    mask = (1 << size) - 1;
    return (val >> offset) & mask;
} else {
    return val;
}
```

https://github.com/diffkemp/diffkemp/ The project was supported by Red Hat Research. xnecas27@stud.fit.vutbr.cz