Cross Translation Unit Test Case Reduction

Réka Kovács / rekanikolett@gmail.com
Eötvös Loránd University / Ericsson Hungary

Test Case Reduction

big file with property of interest
(e.g. triggers crash)

small file with the same
property of interest

Delta Debugging:

remove contiguous regions from the file, test & repeat

Generalized Delta Debugging: C-Reduce

https://embed.cs.utah.edu/creduce/

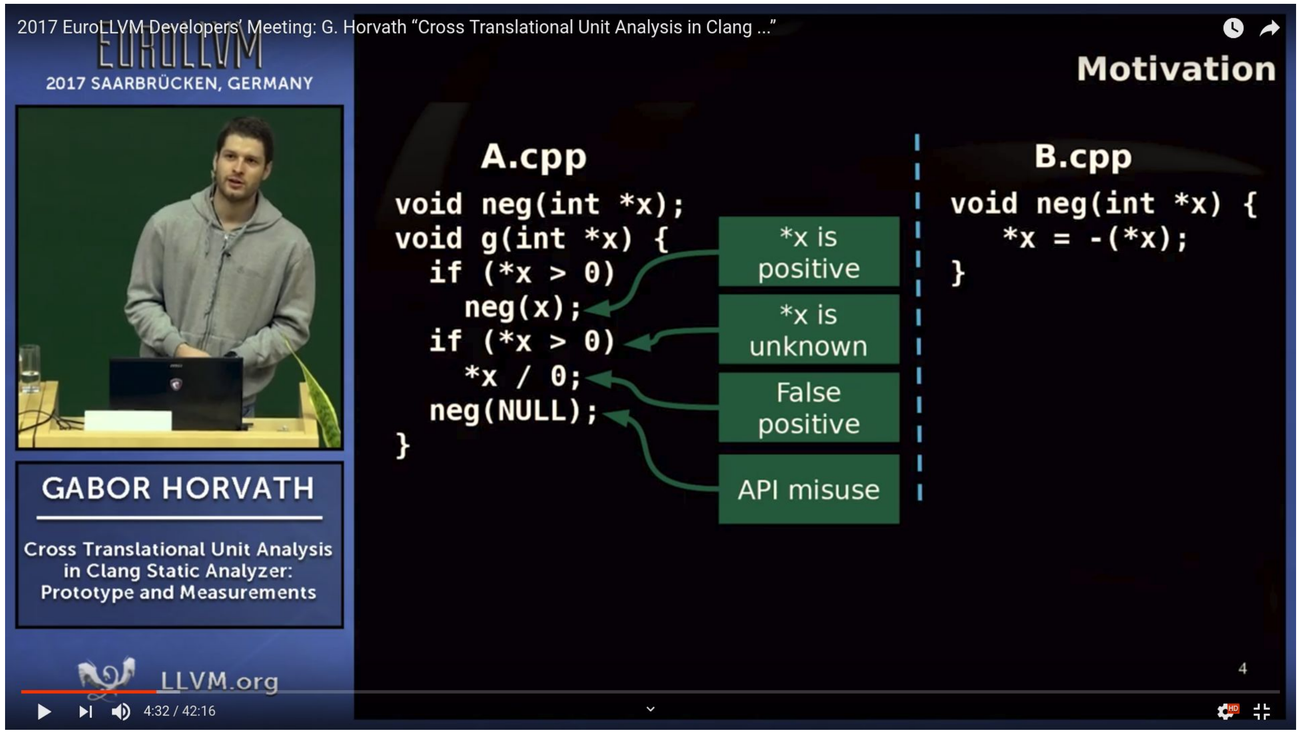
“Compiler-like” transformations (~74 of them): **Clang-Delta**
e.g. remove a parameter from a function, move a parameter to a global variable, scalar replacement of aggregates, etc.

Works on one translation unit at a time

Cross Translation Unit Analysis in the Clang Static Analyzer

llvm.org/devmtg/2017-03/

Developers need minimal tests for crashes and bugs that range **across TU boundaries**



Cross TU Analysis: Importing ASTs

You have **72 preprocessed files** with ~100 000 average LOC

Cross TU Analysis: Importing ASTs

You have **72 preprocessed files** with ~100 000 average LOC

Cross TU analysis crashes

Cross TU Analysis: Importing ASTs

You have **72 preprocessed files** with ~100 000 average LOC

Cross TU analysis crashes

Find the bug!

Reduction Across Translation Units

b.cpp

```
void f(int);

int main() {
  f(5);
}
```

a.cpp

```
void f(int) {
  __builtin_trap();
}
```

```
$ clang++ a.cpp b.cpp
$ ./a.out
Illegal instruction (core dumped)
```

Reduction Across Translation Units

b.cpp

```
void f(int);

int main() {
  f(5);
}
```

a.cpp

```
void f(int) {
  __builtin_trap();
}
```

```
$ clang_delta --transformation=param-to-global a.cpp
```

a.cpp

```
void f(void) {
  __builtin_trap();
}
```

Reduction Across Translation Units

b.cpp

```
void f(int);

int main() {
  f(5);
}
```

a.cpp

```
void f(void) {
  __builtin_trap();
}
```

```
$ clang++ a.cpp b.cpp
/tmp/b-ef5998.o: In function `main':
b.cpp:(.text+0xa): undefined reference to
`f(int)'
clang-8: error: linker command failed with
exit code 1 (use -v to see invocation)
```

Reduction Across Translation Units

b.cpp

```
void f(int);

int main() {
  f(5);
}
```

a.cpp

```
void f(void) {
  __builtin_trap();
}
```

We need to do **the same** transformation on the other file

What is **the same** transformation?

Reduction Across Translation Units

b.cpp

```
void f(int);

int main() {
  f(5);
}
```

Transformation: param-to-global
Available instances: 0

a.cpp

1

```
void f(int) {
  __builtin_trap();
}
```

Transformation: param-to-global
Available instances: 1

Clang-Delta has no notion of **the same** transformation across files

It works with a **counter** of available transformation instances

Reduction Across Translation Units

b.cpp

```
void f(int);

int main() {
  f(5);
}
```

Transformation: param-to-global
Available instances: 0

a.cpp

1

```
void f(int) {
  __builtin_trap();
}
```

Transformation: param-to-global
Available instances: 1

param-to-global **would** handle
all uses of f() if they were in one TU

Unified Symbol Resolution (USR) ?

Thanks!

Would this be useful to you?
Get in touch!

Réka Kovács / rekanikolett@gmail.com