Project Group PhASIO
Input/Output for PhASAR

•••

27 January 2020

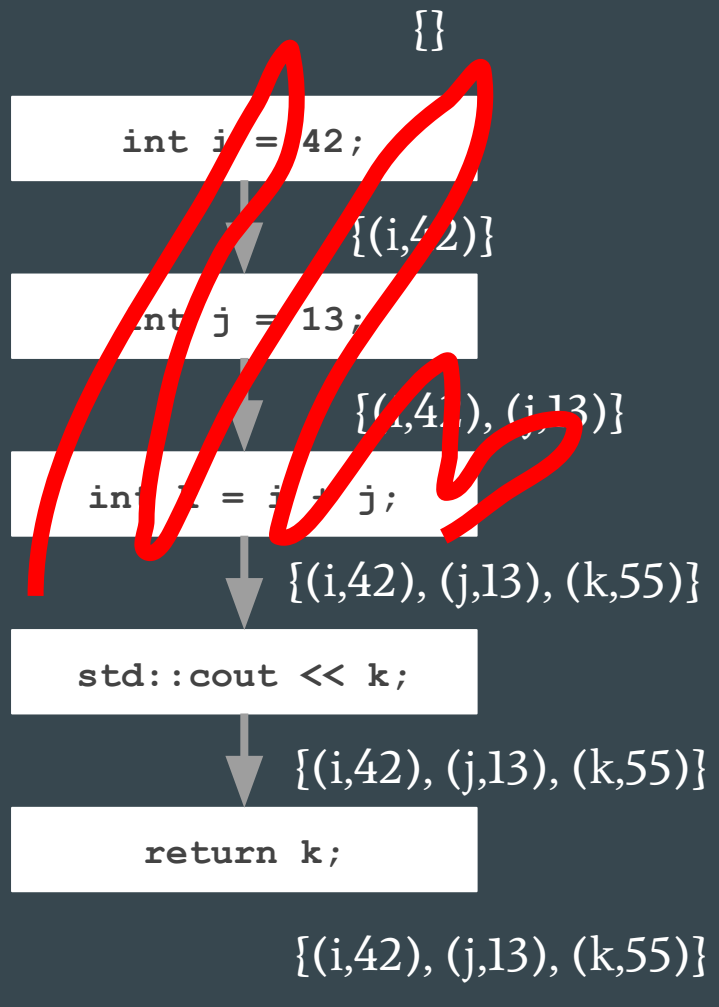
Martin Mory, Philipp Schubert

PhASAR

Static Data-Flow Analysis: Example

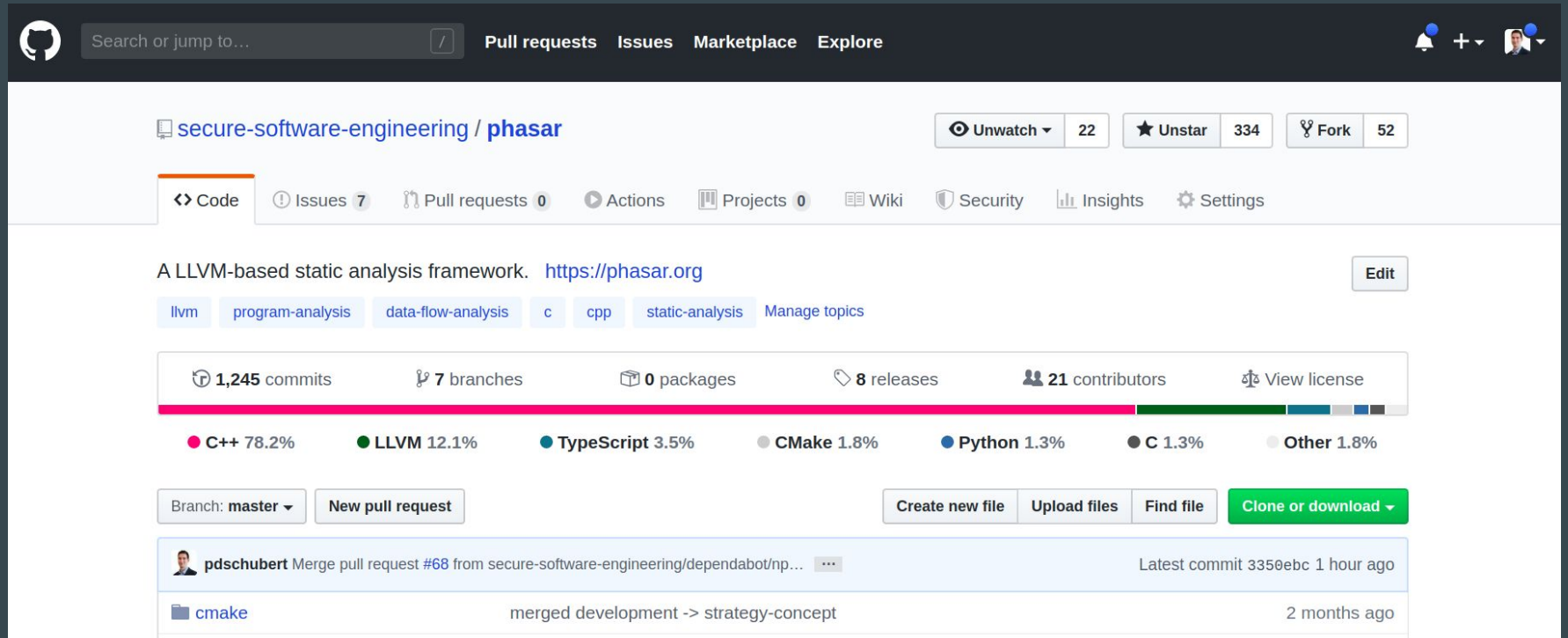
```
#include <iostream>

int main() {

    int i = 42;

    int j = 13;

    int k = i + j;

    std::cout << k;

    return k;

}
```

```
#include <iostream>

int main() {

    std::cout << 55;

    return 55;

}
```



What is PhASAR?

- LLVM-based static analysis framework targeting C/C++



What is PhASAR?

- State-of-the-art static analysis infrastructure
- PhASAR caught attention from static analysis researchers and practitioners

Static Analysis: Workflow

$$
\begin{aligned}
[\![x \leftarrow y]\!](\langle v,t\rangle) &\triangleq \begin{cases} \{\langle x,t\rangle, \langle y,t\rangle\} & \text{if } v = y \\ \{\langle v,t\rangle\} & \text{if } v \neq y \text{ and } v \neq x \\ \emptyset & \text{if } v \neq y \text{ and } v = x \end{cases} \\
[\![y.f \leftarrow x]\!](\langle v,t\rangle) &\triangleq \{\langle v,t\rangle\} \\
[\![x \leftarrow \textbf{null}|\textbf{new } T|y.f]\!](\langle v,t\rangle) &\triangleq \begin{cases} \{\langle v,t\rangle\} & \text{if } v \neq x \\ \emptyset & \text{otherwise} \end{cases} \\
[\![x \leftarrow \textbf{new } T]\!](\mathbf{0}) &\triangleq \{\langle x,T\rangle\} \\
[\![x \leftarrow y.f]\!](\mathbf{0}) &\triangleq \{\langle x,c\rangle : c \in \text{implClasses}(type(f))\} \\
[\![x \leftarrow (T)y]\!](\langle v,t\rangle) &\triangleq \bigcup_{c \in \text{implClasses}(T)} cast(x,y,c)(\langle v,t\rangle) \\
cast(x,y,t_2)(\langle v,t_1\rangle) &\triangleq \begin{cases} \{\langle v,t_1\rangle\} & \text{if } v \neq x \text{ and } v \neq y \\ \emptyset & \text{if } v = x \text{ and } v \neq y \\ \{\langle x,t_1\rangle, \langle y,t_1\rangle\} & \text{if } v = y \text{ and } t_1 <: t_2 \\ \{\langle x,t_2\rangle, \langle y,t_2\rangle\} & \text{if } v = y \text{ and } t_2 <: t_1 \\ \emptyset & \text{if } v = y \text{ and } t_1 \text{ and } t_2 \text{ are unrelated} \end{cases} \\
[\![s]\!](\mathbf{0}) &\triangleq \emptyset \text{ if } s \neq x \leftarrow y.f \text{ and } s \neq x \leftarrow \textbf{new}
\end{aligned}
$$

What About the Results?

IO is Simple, Right?

- Static analysis is not one algorithm
  - Multiple data-flow solvers
  - Concrete client analysis descriptions
  - Parameterizable helper analyses (call-graph, pointer, etc.)
- Source code vs. intermediate representation (IR)
    - Non-trivial differences
- LLVM is a record-based compiler
    - Working on pointers all the time
    - Mapping needs a lot of care (especially when reading inputs)

Project Overview

- Goal: design and implement flexible IO capabilities that allow for
  - fast exchange of static analysis information (between different tools)
  - flexible usage of analysis tools
  - solving tasks like.: “PhASAR, compute a type state analysis using this call-graph in format XYZ!”
- Use suitable formats
  - PhASAR’s json
  - SARIF (Static Analysis Interchange Format)
  - etc.

Project Overview

- Your tasks:
  - Get familiar with involved concepts (compilers, static analysis, formats, etc.)
  - Design input/output components
  - Develop the components in smaller teams on Github
  - Ensure usability of tooling
  - Evaluate on real-world programs
  - IDE integration using LSP (Language Server Protocol)

Project Setting

Number of students: 6 - 8

Requirements

- Basic C++ skills
  - (C++ course SS20, Friday 14-16/18 in C2)
- Communication skills

Beneficial

- Knowledge of good software design and efficient programming
- Knowledge of static analysis, compilers
  - DECA course
- Knowledge of technologies such as: Git, CMake, STL, etc.

Outcome

- Direct contribution to an open-source research project
- Deepening understanding of program analysis, programming languages and compilers
- Excellent C++ skills

Interested?

Talk to us after the presentation.

Martin.Mory@upb.de
Philipp.Schubert@upb.de