

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;
```

```
}
```

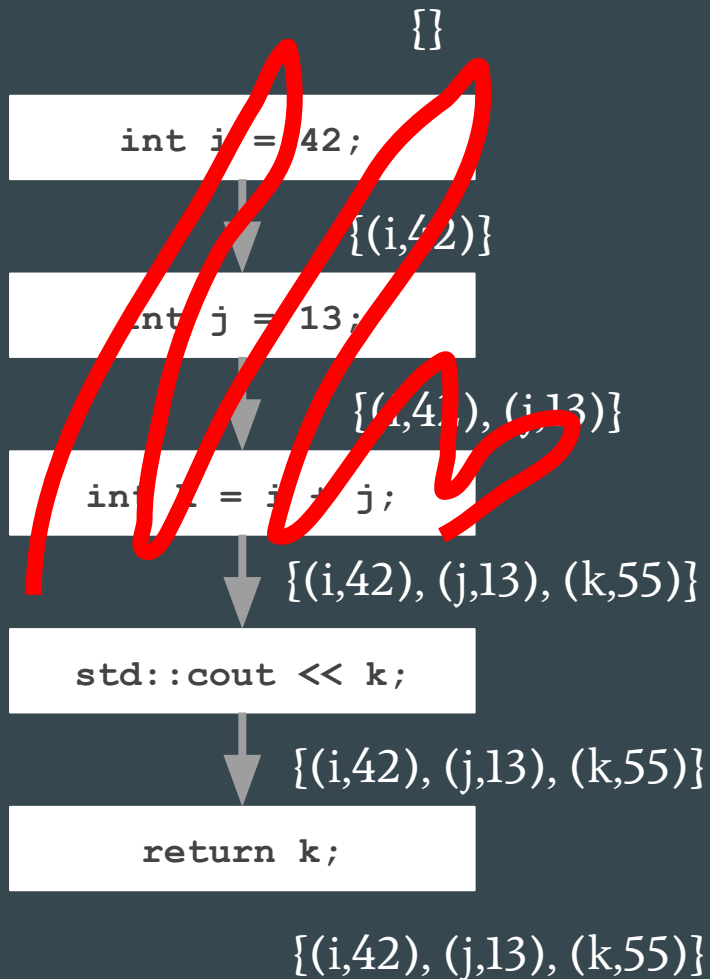
```
#include <iostream>
```

```
int main() {
```

```
    std::cout << 55;
```

```
    return 55;
```

```
}
```



What is PhASAR?

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

The screenshot shows the GitHub repository page for PhASAR. At the top, there is a search bar and navigation links for Pull requests, Issues, Marketplace, and Explore. The repository name is 'secure-software-engineering / phasar'. Below this, there are buttons for Unwatch (22), Unstar (334), and Fork (52). The main navigation bar includes Code, Issues (7), Pull requests (0), Actions, Projects (0), Wiki, Security, Insights, and Settings. The repository description is 'A LLVM-based static analysis framework.' with a link to 'https://phasar.org'. Below the description are tags for 'llvm', 'program-analysis', 'data-flow-analysis', 'c', 'cpp', and 'static-analysis'. A statistics bar shows 1,245 commits, 7 branches, 0 packages, 8 releases, and 21 contributors. A language usage chart shows: C++ 78.2%, LLVM 12.1%, TypeScript 3.5%, CMake 1.8%, Python 1.3%, C 1.3%, and Other 1.8%. At the bottom, there are buttons for 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. A recent commit by 'pdschubert' is shown, merging pull request #68 from 'secure-software-engineering/dependabot/np...' with the latest commit '3350ebc' from 1 hour ago. Below that, a commit for 'cmake' is shown, merged from 'development' to 'strategy-concept' 2 months ago.

What is PhASAR?

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



The screenshot shows the PhASAR website homepage. At the top left is the PhASAR logo, which consists of the word 'PhASAR' in a stylized font where the 'A' is a red circle with a white 'S' inside. To the right of the logo is the text 'PHASAR.ORG' in bold, followed by 'A LLVM-based static analysis framework' in a smaller font. Below the header is a navigation menu with links for 'Phasar', 'Installation', 'Tutorial', 'Download', 'Contribute', 'People', 'Contact', and 'Legal Notice'. The main content area is titled 'POSTS' and features a post dated '11. NOVEMBER 2019' with the title 'Ship Your Analyses Using Docker'. Below the title is a large image of a shipping port at night, with stacks of colorful containers and a ship in the background. To the right of the post is a search bar with the text 'Search ...' and a magnifying glass icon. Below the search bar is a section titled 'RECENT POSTS' which lists three posts: 'Ship Your Analyses Using Docker', 'Support for LLVM 9.0.0', and 'You Can't Fight What You Can't See: Emitting Reports'. The last post in the list is 'Support for LLVM 8.0.0'.

PhASAR PHASAR.ORG
A LLVM-based static analysis framework

[Phasar](#) [Installation](#) [Tutorial](#) [Download](#) [Contribute](#) [People](#) [Contact](#) [Legal Notice](#)

POSTS

11. NOVEMBER 2019

Ship Your Analyses Using Docker

SEARCH ...

RECENT POSTS

- Ship Your Analyses Using Docker
- Support for LLVM 9.0.0
- You Can't Fight What You Can't See: Emitting Reports
- Support for LLVM 8.0.0

Static Analysis: Workflow



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

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

PHASAR

Interested?

**Talk to us after the
presentation.**

Martin.Mory@upb.de

Philipp.Schubert@upb.de

