STATIC ANALYSIS WITH SOOT AND PHASAR

Eric Bodden and Philipp Schubert

SOAP 2020 | 15.06.2020

Fraunhofer IEM

HEINZ NIXDORF INSTITUT
UNIVERSITÄT PADERBORN

AGENDA

1. A few words about us
2. Soot and Phasar
   a) General functionality
   b) Some internals
   c) How to use the frameworks
   d) Future development
3. Contributing

Secure Software Engineering Group

Eric Bodden

Professor for Software Engineering
at Heinz Nixdorf Institute

Director for Software Engineering and
IT-Security at Frauhofer IEM

eric.bodden@upb.de

@profbodden

Philipp Schubert

Ph.D. student, Software Engineering at
Heinz Nixdorf Institute and Fraunhofer IEM

philipp.schubert@upb.de

HEINZ NIXDORF INSTITUT
UNIVERSITÄT PADERBORN

Fraunhofer
IEM


Fraunhofer
IEM
HEINZ NIXDORF INSTITUT
UNIVERSITÄT PADERBORN

Soot and Phasar

A brief history

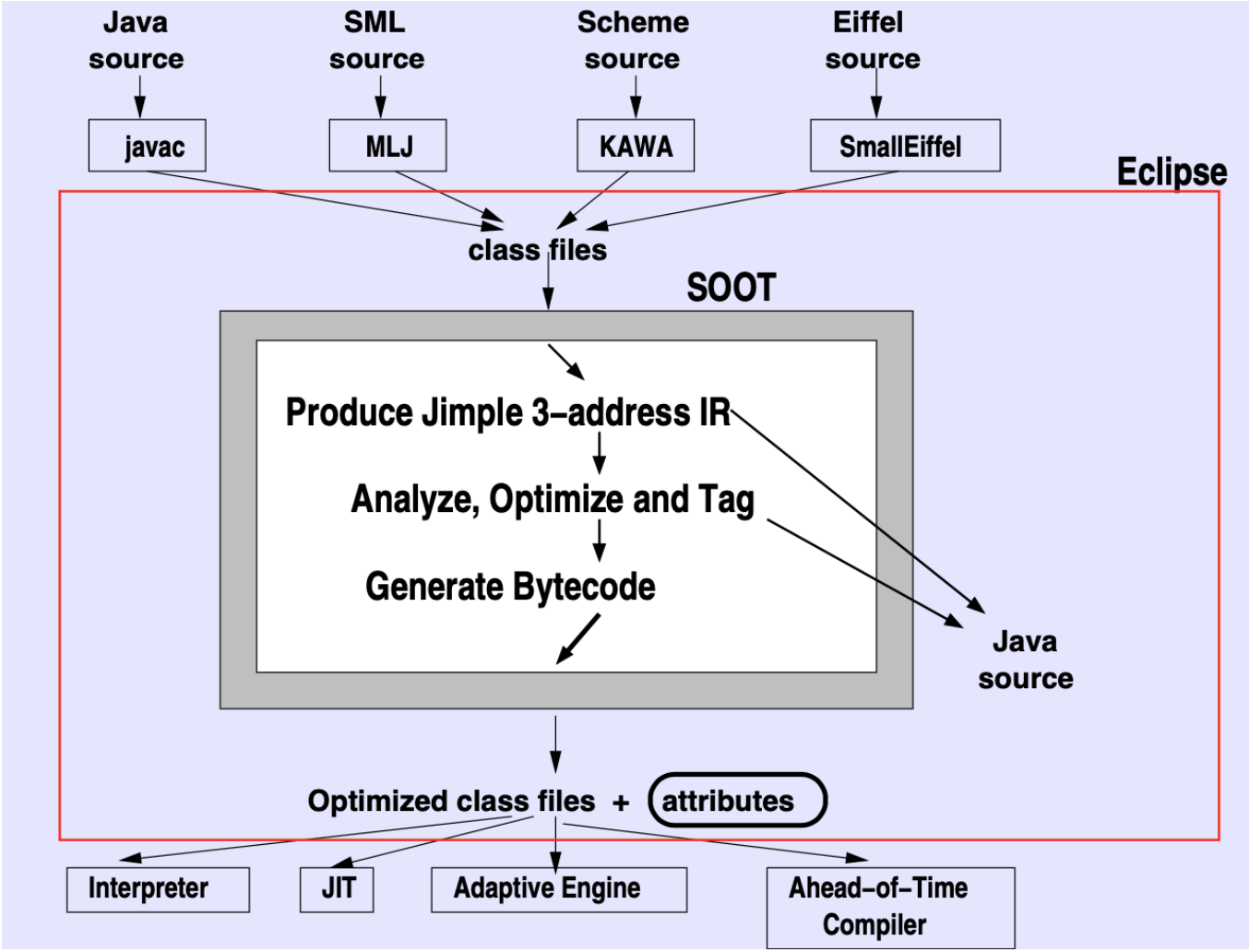
- Established around 1999,
  by Sable Research group @ McGill
- Tutorial at PLDI 2003
- Support for Java, Android (since 2011)
- Large user base, >1.300 citations
  118 👀, 1.400 ☆ on Github
- Several commercial deployments
- LGPL-2.1 licence

https://www.sodafactory.com.au/events/jukebox-thursday-aug1/

Soot and Phasar

A brief history

- Established around 1999,
  by Sable Research group @ McGill
- Tutorial at PLDI 2003
- Support for Java, Android (since 2011)
- Large user base, >1.300 citations
  118 👀, 1.400 ☆ on Github
- Several commercial deployments
- LGPL-2.1 licence



Soot and Phasar

A brief history

- Established around 1999, by Sable Research group
- Tutorial at PLDI 2003
- Support for Java, Android (since 2011)
- Large user base, >1.300 citations
  118 👀, 1.400 ☆ on Github
- Several commercial deployments
- LGPL-2.1 licence

- Established around 2016, by us
- Tutorial at PLDI 2018
- Support for LLVM IR (particularly C/C++)
- Growing user base
  26 👀, 391 ☆ on Github
- Some commercial engagement
- MIT license

Soot and Phasar

How to access

- https://www.soot-oss.org/ (GitHub)
- Usage documentation in wiki
- Easily obtained or built via Maven
- Supports Java bytecode up to version 14, some support for source code
- Code analysis and transformation

- https://phasar.org
- https://github.com/secure-software-engineering/phasar
- Usage documentation in wiki
- Built via CMake
- Supports C-like languages including C/C++
- Code analysis and transformation

Soot and Phasar

Introductory papers

Vallée-Rai et al.: **Soot - a Java bytecode optimization framework.** CASCON'99

Patrick Lam et al. **The Soot framework for Java program analysis: a retrospective.** CETUS 2011

Vallée-Rai and Laurie Hendren: **Jimple: Simplifying Java Bytecode for Analyses and Transformations.** Sable TR 1998-4

Schubert et al.: **Static Analysis for C++ with Phasar.** PLDI'18 tutorial

Schubert et al.: **PhASAR: An Inter-procedural Static Analysis Framework for C/C++.** TACAS'19

Schubert et al.: **Know Your Analysis: How Instrumentation Aids Understanding Static Analysis.** SOAP'19

Soot and Phasar

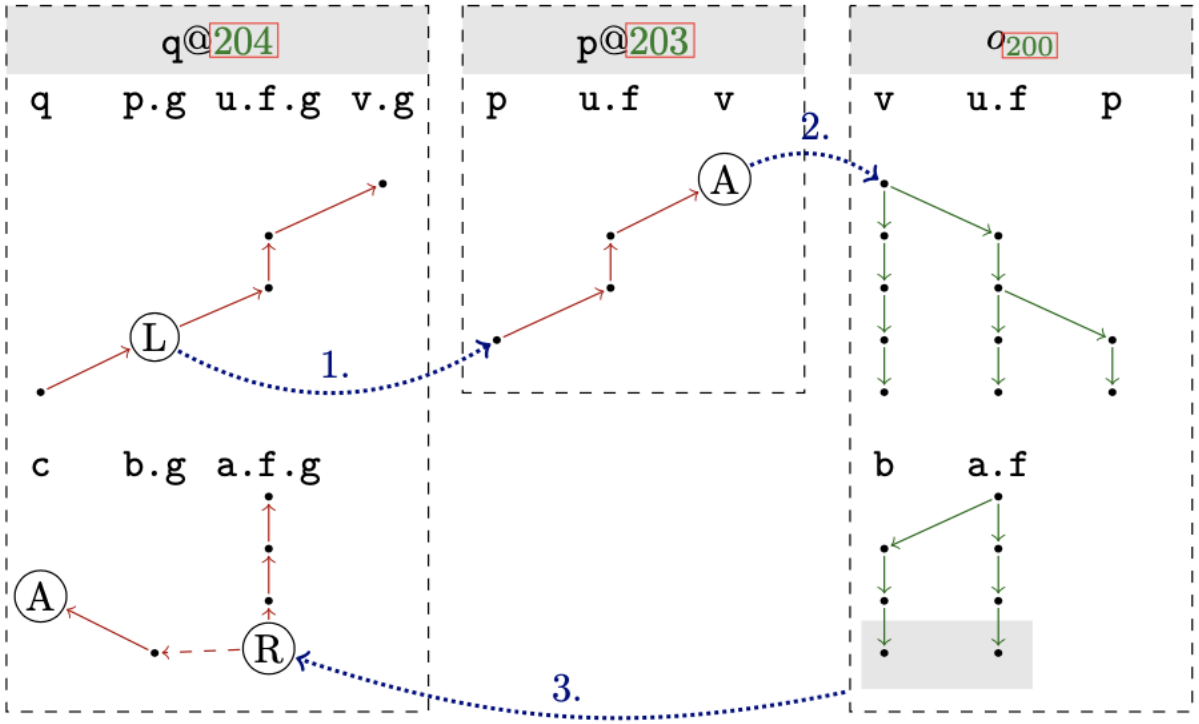
Core functionalities

Intermediate 3-address-code representation	
Jimple, high level	LLVM IR, low level, SSA; other IRs possible
Call-graph analysis	
Spark: CHA, RTA, VTA, Andersen, ..., FlowDroid	CHA, RTA, DTA, Andersen, Steensgaard
Points-to analysis	
Andersen-style, Refinement-based (PLDI‘06) Boomerang (ECOOP‘16)	LLVM points-to infrastructure (including Andersen and Steensgaard-style analyses)
Inter-procedural data-flow solvers	
IFDS/IDE (through Heros), FlowDroid, IDE*al*, Synchr. Pushdown Systems (SPDS)	IFDS/IDE, Call-strings, WPDS (via WALi-OpenNWA)

Latest Solver: Synchronized Pushdown System (SPDS)

Demand-driven flow-, field- and context-sensitive

```
198 foo(){
199   u = new;
200   v = new;
201   u.f = v;
202   setG(u);
203   p = u.f;
204   q = p.g;
205 }
206 setG(a){
207   b = a.f;
208   c = new;
209   b.g = c;
210 }
```



no k-limiting!

∞-context-sensitive

∞-field-sensitive

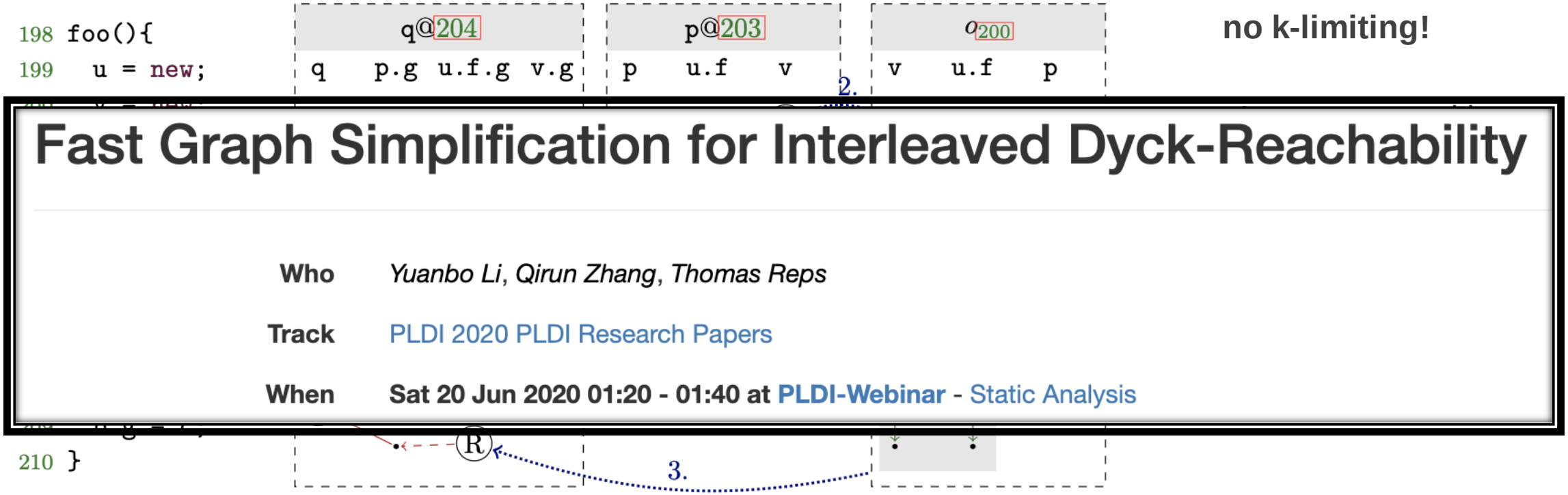
... both for pointer and client analysis!

Johannes Späth: **Synchronized Pushdown Systems for Pointer and Data-Flow Analysis**, PhD thesis, Universität Paderborn, 2019.

Johannes Späth, Karim Ali, Eric Bodden: **Context-, Flow-, and Field-sensitive Data-flow Analysis Using Synchronized Pushdown Systems**, In Proceedings of the ACM SIGPLAN Symposium on Principles of Programming Languages, pages 48:1–48:29, 3(POPL), 2019.

Latest Solver: Synchronized Pushdown System (SPDS)

Demand-driven flow-, field- and context-sensitive



Johannes Späth: **Synchronized Pushdown Systems for Pointer and Data-Flow Analysis**, PhD thesis, Universität Paderborn, 2019.

Johannes Späth, Karim Ali, Eric Bodden: **Context-, Flow-, and Field-sensitive Data-flow Analysis Using Synchronized Pushdown Systems**, In Proceedings of the ACM SIGPLAN Symposium on Principles of Programming Languages, pages 48:1–48:29, 3(POPL), 2019.

Soot and Phasar

Interaction and usages

- Provides core functionalities
- Generic data-flow solvers as extensions
- Java dependency
  - Can be included using *Maven*, Gradle, SBT, etc.
- Command-line tool
- Get help through mailing list and issue tracker

- Monorepo organized using CMake
- Usages
  - (Sub)libraries
  - Plugins
  - *phasar-llvm* command-line tool
- Find help on Slack phasar.slack.com

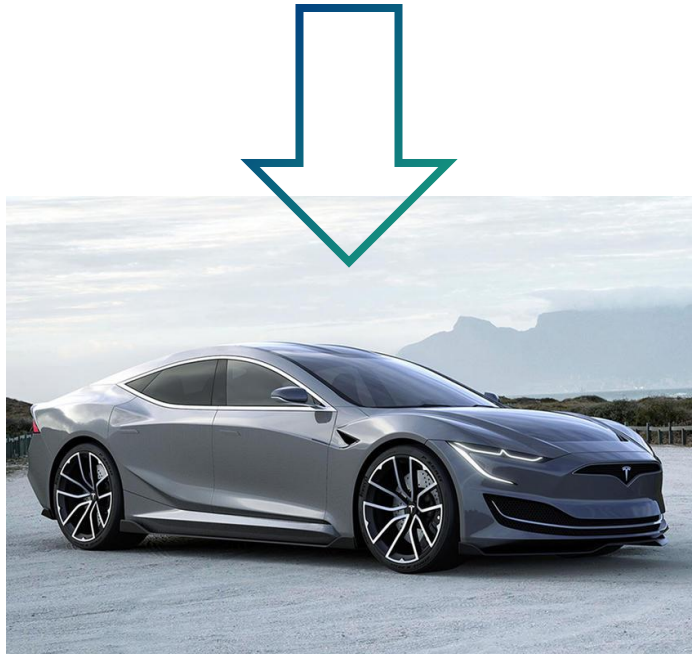
Soot and Phasar

Future development

Funded by
DFG Deutsche Forschungsgemeinschaft
German Research Foundation

FutureSoot DFG project

- Completely new architecture (no singletons)
- Support for multiple “scenes”, comparative and incremental code analysis
- More modularity, less legacy, better testability
- IR akin to Jimple but optimized for fast access
- First release hopefully still this year
- Upcoming: workshop with interested stakeholders



Soot and Phasar

Future development

https://images.app.goo.gl/ShtuzjJSDG6aeeJr9

Technical improvements

- Completely revised analysis model
- Improved usability
- Performance improvements

Old dogs new tricks: improve on SPDS

- Add support for strong updates
- Fine tune algorithmic and technical details
- Overcome C/C++ specific difficulties
- Evaluation on large production software

Using Soot and PhASAR

Contributions and Collaborations

- Static Analysis is hard
- Maintaining tools is fun but a lot of work
- We welcome any contribution
  - Report bugs
  - Fix bugs `good first issue`
  - Contribute features
- Become a committer!

Questions?