

# T.J. Watson Libraries for Analysis (WALA)

<http://d3s.mff.cuni.cz>

Department of  
Distributed and  
Dependable  
Systems



*Pavel Parízek*



CHARLES UNIVERSITY IN PRAGUE

faculty of mathematics and physics

- Static analysis library
  - Open source (SVN, Git, Wiki)
- Languages
  - Java (source, bytecode), JavaScript
- Home page
  - <http://wala.sourceforge.net/>

# How to get WALA

- Download and unpack somewhere
  - [http://d3s.mff.cuni.cz/teaching/program\\_analysis\\_verification/files/wala-examples.zip](http://d3s.mff.cuni.cz/teaching/program_analysis_verification/files/wala-examples.zip)
- Content
  - Subdirectory `wala`
    - binary (jar file), source code, configuration, exclusion file (ignored libraries)
  - Subdirectory `src`
    - two example analyses, two simple programs
  - Ant build script (`build.xml`)

# Example: static analyses

- Directory `src/analysis`
  - Merge operator: bit vector intersection
  - Analysis facts: `FieldID`, `ProgramPoint`
  - Several utility methods: `WALAUtils`
- Two static analyses
  - Locked local variables
  - Future field accesses

- SSA: static single assignment form
- IR: intermediate representation
- Characteristics
  - Exactly one assignment to each local variable
  - Multiple assignments in source code → multiple incarnations in the SSA IR
  - Unique value numbers for variables (expressions)
  - Value numbers: `this`, parameters, local variables

# Example: locked local variables

- Intra-procedural forward analysis
  - For each method reachable in the call graph
- Flow-sensitive context-insensitive
- Analysis facts: SSA value numbers
- Merge operator: set intersection
- Relevant instructions (SSA)
  - Monitor enter
  - Monitor exit

# Inter-procedural CFG

- Caller method
  - Nodes: call, return
- Callee method
  - Nodes: entry, exit
- ICFG edges
  - call  $\rightarrow$  entry
  - exit  $\rightarrow$  return
  - call  $\rightarrow$  return

# Example: future field accesses

- Inter-procedural backward analysis
  - Over the whole program inter-procedural CFG
- Flow-sensitive context-insensitive
- Merge operator: set union (“may”)
- Bytecode instructions
  - Field access (read, write)
- Running
  - Ant: `run.example.fieldaccess`
  - Input: `example.WholeProgram`



# Documentation

- Tutorial

- <http://wala.sourceforge.net/wiki/index.php/Tutorials>

- User guide

- WALA core technical overview
- [http://wala.sourceforge.net/wiki/index.php/Wala.core\\_technical\\_overview](http://wala.sourceforge.net/wiki/index.php/Wala.core_technical_overview)

- API docs

- List of SSA instructions (`com.ibm.wala.ssa`)
- <http://wala.sourceforge.net/javadocs/trunk/>