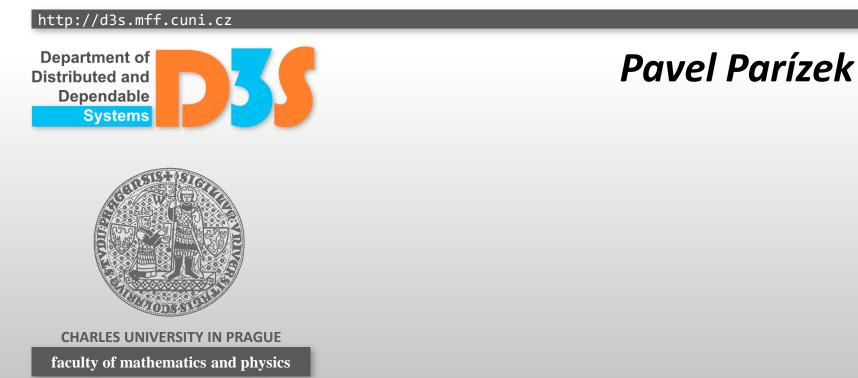
CEGAR



Tools

- Connect to some Linux machine
 - using SSH (Putty)
- Download
 - http://d3s.mff.cuni.cz/teaching/program analysis verification/files/cegar.tgz
- Package contains Linux binaries of
 - BOPPO
 - Model checker for boolean programs
 - SATABS v1.9
 - CEGAR + SAT
 - BLAST v2.5
 - Lazy abstraction
 - Examples
 - Some taken from tutorials created by authors of respective tools

SATABS

- Verification tool for C and C++ programs
 - Based on CEGAR
 - Uses a SAT solver
- Key features
 - Variables represented as bit vectors (binary level)
 - Computer arithmetic (overflow, bit operators, ...)
- Developed at ETH Zurich & Carnegie Mellon Uni
- <u>http://www.cprover.org/satabs/</u>
- Source code and binaries freely available
 - Platforms: Windows, Linux, Mac OS

3

SATABS: example 1

- Set environment variables
 - .../cegar-cfg.sh
- Make all binaries executable chmod u+x <file>
- Run SATABS

cd examples/ex01
satabs --modelchecker boppo main.c

- Tasks
 - Change the program in order to 1) violate the assertion and 2) force SATABS to make more iterations

istributed and

SATABS: example 2

- Subject: a dummy Linux 2.0 device driver
- Running
 - cd examples/ex02
 - satabs --modelchecker boppo spec.c driver.c
- Tasks
 - Inspect the source code and header files
 - Fill in the missing parts of the testing harness
 - See the TODO mark in the file spec.c
 - open has to be called (with success) before read
 - release has to be called before exiting
 - Use SATABS to verify the program (or to find bugs)
 - Hint
 - Use nondet_uint() with __CPROVER_assume()

Distributed and

- Key feature: lazy predicate abstraction
- Developed at UC Berkeley & EPFL (Lausanne)
- <u>http://mtc.epfl.ch/software-tools/blast/</u>

- Obsoleted by CPAchecker
 - Many advanced features and optimizations

- Make necessary binaries executable
 pblast.opt, spec.opt, csisat, Simplify
- How to run BLAST cd examples/ex03 gcc -E -I . tut1.c > tut1.i pblast.opt -main foo tut1.i
- Tasks
 - Correct the program and verify using BLAST

Distributed and

BLAST: example 4

- BLAST property specification language
- lock.spc
 - Defines correct locking & unlocking
- How to run BLAST with custom property spec.opt lock.spc tut2.c pblast.opt instrumented.c
- Tasks
 - Look at the instrumented code
 - Try to find and correct the bug

BLAST: example 5

- Simple file wrapper for reading lines
 - reader.{c,h} file wrapper
 - error_handling.h macros
 - main.c very simple test case

- Tasks
 - Define your own property that captures locking & unlocking discipline (hint: reuse ex04)
 - Find all property violations and fix the program

istributed and

- Try to run BLAST and SATABS on your own programs in C/C++
 - Insert some assertions to your code (if necessary)



CPAchecker

- Modern successor of BLAST
 - Still under development
- Input: programs in C
- Advantages
 - Highly configurable
 - abstraction, merging data from control-flow paths
 - More user- friendly
- Web: <u>https://cpachecker.sosy-lab.org/</u>

