DIVINE: Analysis of C++ Programs

Vladimír Štill

ParaDiSe
Parallel & Distributed
Systems Laboratory

Masaryk University

Brno, Czech Republic

22nd September 2017

DIVINE: Analysis of C++ Programs

- DIVINE is an analyser for C and C++ programs
- focus on language features
- built on top of the LLVM infrastructure
- tightly integrated compiler

Current Status: Publications

waiting in journal review process:

- Petr Ročkai, Vladimír Štill, Ivana Černá, Jiří Barnat:
  DiVM: Model Checking with LLVM and Graph Memory

Current Status: Publications

waiting in journal review process:

- Petr Ročkai, Vladimír Štill, Ivana Černá, Jiří Barnat:
  DiVM: Model Checking with LLVM and Graph Memory

freshly published:

- Vladimír Štill, Petr Ročkai, Jiří Barnat:
  Using Off-the-Shelf Exception Support Components in C++ Verification

Current Status: Publications

waiting in journal review process:

- Petr Ročkai, Vladimír Štill, Ivana Černá, Jiří Barnat:
  DiVM: Model Checking with LLVM and Graph Memory

freshly published:

- Vladimír Štill, Petr Ročkai, Jiří Barnat:
  Using Off-the-Shelf Exception Support Components in C++ Verification
- **Model Checking of C and C++ with DIVINE 4**

Current Status: Publications

waiting in journal review process:

- Petr Ročkai, Vladimír Štill, Ivana Černá, Jiří Barnat:
  DiVM: Model Checking with LLVM and Graph Memory

freshly published:

- Vladimír Štill, Petr Ročkai, Jiří Barnat:
  Using Off-the-Shelf Exception Support Components in C++ Verification
- **Model Checking of C and C++ with DIVINE 4**
- Katarína Kejstová, Petr Ročkai, Jiří Barnat:
  From Model Checking to Runtime Verification and Back

Current Status: Publications

waiting in journal review process:

- Petr Ročkai, Vladimír Štill, Ivana Černá, Jiří Barnat:
  DiVM: Model Checking with LLVM and Graph Memory

freshly published:

- Vladimír Štill, Petr Ročkai, Jiří Barnat:
  Using Off-the-Shelf Exception Support Components in C++ Verification
- **Model Checking of C and C++ with DIVINE 4**
- Katarína Kejstová, Petr Ročkai, Jiří Barnat:
  From Model Checking to Runtime Verification and Back

reworking:

- Petr Ročkai, Jiří Barnat: **A Simulator for LLVM Bitcode**

Current Status: New Features I

DIVINE Operating System (DiOS)

- modularized for more flexibility and configurability
- support for processes (`fork`)

Current Status: New Features I

DIVINE Operating System (DiOS)

- modularized for more flexibility and configurability
- support for processes (`fork`)

Symbolic Data

- symbolization of data structures
- multiple data domains in the same program
- refactoring and cleaning

Current Status: New Features I

DIVINE Operating System (DiOS)

- modularized for more flexibility and configurability
- support for processes (`fork`)

Symbolic Data

- symbolization of data structures
- multiple data domains in the same program
- refactoring and cleaning

Relaxed Memory Models

- resurrected and refactored implementation
- work on efficiency

Current Status: New Features II

Simulator

- multiple usability improvements

Current Status: New Features II

Simulator

- multiple usability improvements

Algorithms

- more robust counterexample format
- `divine run` can output stream of executed instructions

Current Status: New Features II

Simulator

- multiple usability improvements

Algorithms

- more robust counterexample format
- `divine run` can output stream of executed instructions

Core

- improved state space reduction

Planned Publications

- **DiOS: A Lightweight Approach to Verifying POSIX-Based Programs**

Planned Publications

- **DiOS: A Lightweight Approach to Verifying POSIX-Based Programs**
- **A Semi-Dynamic State Space Reduction for C/C++ Programs**

Planned Publications

- **DiOS: A Lightweight Approach to Verifying POSIX-Based Programs**
- **A Semi-Dynamic State Space Reduction for C/C++ Programs**
- **Symbolic Computation via Program Transformations**

Planned Publications

- **DiOS: A Lightweight Approach to Verifying POSIX-Based Programs**
- **A Semi-Dynamic State Space Reduction for C/C++ Programs**
- **Symbolic Computation via Program Transformations**
- **Verification of Concurrent C++ Programs with Realistic Memory Models**

Runtime Verification

- currently we can record and replay syscall interatction

Runtime Verification

- currently we can record and replay syscall interatction
- replay can explore further runs

Runtime Verification

- currently we can record and replay syscall interatction
- replay can explore further runs
- but syscall ordering has to match

Runtime Verification

- currently we can record and replay syscall interatction
- replay can explore further runs
- but syscall ordering has to match
- in future we want to allow reordering of independent syscalls

Runtime Verification

- currently we can record and replay syscall interatction
- replay can explore further runs
- but syscall ordering has to match
- in future we want to allow reordering of independent syscalls
- also possibility to replace replies of some syscalls with symbolic/nondeterministic values

Relaxed Memory Models: Motivation

What is relaxed memory model?

- describes how memory operations propagate between processors

Relaxed Memory Models: Motivation

What is relaxed memory model?

- describes how memory operations propagate between processors
- effects of cache, out-of-order execution

Relaxed Memory Models: Motivation

What is relaxed memory model?

- describes how memory operations propagate between processors
- effects of cache, out-of-order execution
- depends on architecture

Relaxed Memory Models: Motivation

What is relaxed memory model?

- describes how memory operations propagate between processors
- effects of cache, out-of-order execution
- depends on architecture
    - Intel x86: store buffering
    - ARM/POWER: almost arbitrary reordering of loads and stores

Relaxed Memory Models: Motivation

What is relaxed memory model?

- describes how memory operations propagate between processors
- effects of cache, out-of-order execution
- depends on architecture
    - Intel `x86`: store buffering
    - ARM/POWER: almost arbitrary reordering of loads and stores
- significantly increases theoretical complexity of verification

Relaxed Memory Models

- rest of my PhD work
- thesis proposal submitted

Relaxed Memory Models

- rest of my PhD work
- thesis proposal submitted
- focus on mitigation of state space explosion

Relaxed Memory Models

- rest of my PhD work
- thesis proposal submitted
- focus on mitigation of state space explosion
  - a lot of equivalent interleavings with conventional methods
  - opportunity for static analysis, heuristics

Relaxed Memory Models

- rest of my PhD work
- thesis proposal submitted
- focus on mitigation of state space explosion
  - a lot of equivalent interleavings with conventional methods
  - opportunity for static analysis, heuristics
- plans to support multiple memory models

Relaxed Memory Models

- rest of my PhD work
- thesis proposal submitted
- focus on mitigation of state space explosion
  - a lot of equivalent interleavings with conventional methods
  - opportunity for static analysis, heuristics
- plans to support multiple memory models
- completeness of analysis

Symbolic and Abstract Verification: Motivation

- program's behaviour can depend on input values

Symbolic and Abstract Verification: Motivation

- program's behaviour can depend on input values
- need for verification under arbitrary inputs

Symbolic and Abstract Verification: Motivation

- program's behaviour can depend on input values
- need for verification under arbitrary inputs
- cannot be handled explicitly: too many combinations

Symbolic and Abstract Verification

main idea: transform the program to encode symbolic/abstracted data

- Henrich's master's thesis, later PhD

Symbolic and Abstract Verification

main idea: transform the program to encode symbolic/abstracted data

- Henrich's master's thesis, later PhD
- framework for using different abstractions

Symbolic and Abstract Verification

main idea: transform the program to encode symbolic/abstracted data

- Henrich's master's thesis, later PhD
- framework for using different abstractions
- master's thesis in December
  - finalizing the initial implementation
  - testing and benchmarking
  - support for functions, data structures
  - but no dynamic memory and aliased pointers

Symbolic and Abstract Verification

main idea: transform the program to encode symbolic/abstracted data

- Henrich's master's thesis, later PhD
- framework for using different abstractions
- master's thesis in December
  - finalizing the initial implementation
  - testing and benchmarking
  - support for functions, data structures
  - but no dynamic memory and aliased pointers
- planned PhD work
  - full abstraction of programs with memory
  - symbolic memory
  - . . .