Verification and Static Analysis of Software

Introduction
Summer Semester 2017; 20 April, 2017
B. Kaminski, C. Matheja, T. Noll, M. Volk
Software Modeling and Verification Group
RWTH Aachen University

https://moves.rwth-aachen.de/teaching/ss-17/vsas/
Overview

Outline

Overview

Aims of this Seminar

Important Dates

Pointer and Shape Analysis

Advanced Model Checking Techniques

Analysis of Probabilistic Programs

Final Hints
Overview

Formal Methods

Formal methods

- Rigorous, mathematically based techniques for the specification, development, analysis, and verification of software and hardware systems
- Aim at improving correctness, reliability and robustness of such systems
Overview

Formal Methods

Formal methods

- Rigorous, mathematically based techniques for the specification, development, analysis, and verification of software and hardware systems
- Aim at improving correctness, reliability and robustness of such systems

Classifications

- According to design phase
  - specification, implementation, testing, ...
- According to specification formalism
  - source code, process algebras, timed automata, Markov chains, ...
- According to underlying mathematical theories
  - model checking, theorem proving, static analysis, ...

3 of 37 Verification and Static Analysis of Software
Kaminski/Matheja/Noll/Volk
Summer Semester 2017; 20 April, 2017
Overview

Areas Covered in this Seminar

Areas

- **Pointer and Shape Analysis**
  - *Static Program Analysis* (WS 2016/17)
  - *Semantics and Verification of Software* (SS 2015)

- **Advanced Model Checking Techniques**
  - *Advanced Model Checking* (WS 2016/17)
  - *Introduction to Model Checking* (SS 2016)

- **Analysis of Probabilistic Programs**
  - *Probabilistic Programming* (WS 2016/17)
  - *Modelling and Verification of Probabilistic Systems* (WS 2015/16)
Aims of this Seminar

Outline

Overview

Aims of this Seminar

Important Dates

Pointer and Shape Analysis

Advanced Model Checking Techniques

Analysis of Probabilistic Programs

Final Hints
## Aims of this Seminar

### Goals

<table>
<thead>
<tr>
<th>Aims of this seminar</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Independent understanding of a scientific topic</td>
</tr>
<tr>
<td>• Acquiring, reading and understanding scientific literature</td>
</tr>
<tr>
<td>• Writing of your own report on this topic</td>
</tr>
<tr>
<td>• Oral presentation of your results</td>
</tr>
</tbody>
</table>
Aims of this Seminar

Requirements on Report

Your report

- Independent writing of a report of ≈ 15 pages
- Complete set of references to all consulted literature
- Correct citation of important literature
- Plagiarism: taking text blocks (from literature or web) without source indication causes immediate exclusion from this seminar
- Font size 12pt with “standard” page layout
- Language: German or English
- We expect the correct usage of spelling and grammar
  - ≥ 10 errors per page → abortion of correction
- Report template will be made available on seminar web page
Aims of this Seminar

Requirements on Talk

Your talk

- Talk of about 45 (= 40 + 5) minutes
- Focus your talk on the audience
- Descriptive slides:
  - ≤ 15 lines of text
  - use (base) colors in a useful manner
- Language: German or English
- No spelling mistakes please!
- Finish in time. Overtime is bad
- Ask for questions
Aims of this Seminar

Final Preparations

Preparation of your talk

- Setup laptop and projector ahead of time
- Use a (laser) pointer
- Number your slides
- Multiple copies: laptop, USB, web
- Have backup slides ready for expected questions
Important Dates

Outline

Overview

Aims of this Seminar

Important Dates

Pointer and Shape Analysis

Advanced Model Checking Techniques

Analysis of Probabilistic Programs

Final Hints
Important Dates

Deadlines

- 15 May: Detailed outline of report due
- 12 June: Report due
- 3 July: Presentation slides due
- 17 July (?): Seminar

Missing a deadline causes immediate exclusion from the seminar.
## Important Dates

### Deadlines

- 15 May: Detailed outline of report due
- 12 June: Report due
- 3 July: Presentation slides due
- 17 July (?): Seminar

---

Missing a deadline causes **immediate exclusion** from the seminar.
**Important Dates**

**Selecting Your Topic**

**Procedure**

- You obtained a list of topics of this seminar.
- Indicate the preference of your topics (first, second, third).
- Return sheet by **Monday (24 April)** via e-mail/to secretary.
- We do our best to find an adequate topic-student assignment.
  - disclaimer: no guarantee for an optimal solution
- Assignment will be published on web site next week.
- Then also your **supervisor** will be indicated.
Important Dates

Selecting Your Topic

Procedure

- You obtain(ed) a list of topics of this seminar.
- Indicate the preference of your topics (first, second, third).
- Return sheet by Monday (24 April) via e-mail/to secretary.
- We do our best to find an adequate topic-student assignment.
  - disclaimer: no guarantee for an optimal solution
- Assignment will be published on web site next week.
- Then also your supervisor will be indicated.

Withdrawal

- You have up to three weeks to refrain from participating in this seminar.
- Later cancellation (by you or by us) causes a not passed for this seminar and reduces your (three) possibilities by one.
Pointer and Shape Analysis

Outline

Overview

Aims of this Seminar

Important Dates

Pointer and Shape Analysis

Advanced Model Checking Techniques

Analysis of Probabilistic Programs

Final Hints
Pointer and Shape Analysis

Pointer-Related Software Errors

Sequential programming errors

- Dereferencing invalid pointers
- Creation of memory leaks
- Invalidation of data structures
Pointer and Shape Analysis

Pointer-Related Software Errors

Sequential programming errors
- Dereferencing invalid pointers
- Creation of memory leaks
- Invalidation of data structures

Concurrent programming errors
- Deadlocks
- Data races
- ...

https://xkcd.com/371
Problem Analysis: unbounded state spaces with irregular structure

- Infinite data domains
- Dynamic storage (de-)allocation
- Destructive pointer updates
- Recursive procedures
- Dynamic thread creation
### Analysis problem: unbounded state spaces with irregular structure

- Infinite data domains
- Dynamic storage (de-)allocation
- Destructive pointer updates
- Recursive procedures
- Dynamic thread creation

### Solution: abstraction

- Automata-based: regular model checking, forest automata
- Graph-based: graph grammars, graph transformation systems
- Logic-based: shape analysis, separation logic
- Extensions for concurrency
### Pointer and Shape Analysis

### Problems

**Analysis problem: unbounded state spaces with irregular structure**

- Infinite data domains
- Dynamic storage (de-)allocation
- Destructive pointer updates
- Recursive procedures
- Dynamic thread creation

### Solution: abstraction

- Automata-based: regular model checking, forest automata
- Graph-based: graph grammars, graph transformation systems
- Logic-based: shape analysis, separation logic
- Extensions for concurrency
**Pointer and Shape Analysis**

1: Fractional Permissions for Concurrency
2: Symbolic Permission Accounting

---

**Idea**

- Threads acquire/release read and write permissions (fractional values between 0 and 1)
- Partial permissions $0 < p < 1$ for **shared read** access
- Full permission $p = 1$ for **exclusive write** access

**Observations**

- Permission not available $\implies$ (potential) **data race**
- Permissions can always be acquired $\implies$ **data-race freedom**

---

**Here:** two approaches to **symbolically** represent permissions
### 3: Compositional Shape Analysis by Means of Bi-Abduction

#### Terms

- **Shape analysis**: static analysis to discover and verify properties of pointer programs
- **Compositional analysis**: each procedure is analyzed independently of its callers
- **Abduction**: identify part \(? \) of a formula to make implication \( \varphi \star ? \rightarrow \psi \) valid
  - \( \varphi \): assertion at call site
  - \( \psi \): procedure precondition

#### Approach

- Heuristic to solve abduction problem of separation logic
- Use abduction to obtain a compositional shape analysis generating pre/post-conditions for each procedure
- Apply analysis to real-world programs: Linux Kernel, GIMP, Emacs, Sendmail, . . .
- Provides theoretical foundations of a static analyzer called Infer, developed and used by Facebook
4: Amortised Resource Analysis

Example

```c
for (ptr = head; ptr != null; ptr = ptr.next) {
    expensiveOperation(ptr.data);
    ptr = ptr.next;
}
```

What is it all about?

- What is the run-time complexity of this program?
- Resource usage depends on length of list
- Handled nicely by amortised resource analysis
- Use Separation Logic to automatically derive complexity bounds

Main Ideas

- Combine Separation Logic with resources
- \{R\}_{\text{consume}}(R)_{\text{emp}}: “consume R at a given cost”
- Use type system for automated amortized complexity analysis
Advanced Model Checking Techniques

Outline

Overview

Aims of this Seminar

Important Dates

Pointer and Shape Analysis

Advanced Model Checking Techniques

Analysis of Probabilistic Programs

Final Hints
Advanced Model Checking Techniques

Model Checking

```
real system

abstraction semantics

abstract model

finite transition system

specification

finite transition system

requirements

or temporal formula, e.g.,
□(request → ◇enter_crit)

model checker

"does M sat spec hold?"

yes

no
```
1: Counterexample-Guided Abstraction Refinement

- Main problem from model checking: large state spaces
- Idea: only consider abstraction $\text{Abs}(T)$ of system $T$
- Abstraction is over-approximation
- If property is satisfied on $\text{Abs}(T) \implies$ satisfied on $T$
- Otherwise found counterexample
- If also counterexample for $T \implies$ property violated
- Else refine abstraction using counterexample
Advanced Model Checking Techniques

2: Assume-Guarantee Reasoning

- Modular model checking
- Check each module \((M_1, M_2)\) on its own
- Use assumption \(A\) to show property \(P\)

\[
\langle A \rangle M_1 \langle P \rangle, \langle true \rangle M_2 \langle A \rangle \quad \frac{\langle true \rangle M_1 \parallel M_2 \langle P \rangle}{\langle true \rangle M_1 \parallel M_2 \langle P \rangle}
\]

- Idea: iteratively compute assumption \(A_0, A_1, \ldots\) and refine
3: Fairness

\[ \alpha : x := x + 1 \]

\[ x := 0, y := 0 \]

\[ \beta : y := y - 1 \]

- Fairness important when considering multiple processes
- Algorithms for finite state system operate “locally”
- Now algorithm for infinite state systems
4: Bounded Model Checking

- **Bounded model checking (BMC)** is a powerful bug-hunting technique.
- Is applied to hard- and software.
- Its basis is to consider paths up to a certain depth $k$.
- The transition system is encoded as **Boolean formula**.
- Modern SAT solvers are applied to check for counterexamples.
- Generalizations for liveness and arbitrary depths $k$ do exist.
5: Configurable Software Verification

Configurable SW Verification:
- Static Analysis (SA) and Verification reducible to each other
- SA knows generic algorithm for decades
- Won Goedel medal “for their contributions to the development of efficient verification methods and algorithms”

Adjustable Block Encoding
- CEGAR hampered by large programs, especially sequences
- Simplify program by folding sequences
  \[\text{[Beyer et al. 2009]}\]
- Folding until minimality sometimes not very efficient, follow spirit of CPA and make it adjustable
6: IC3

Consider the transition system $\mathcal{M} = (X, I, T)$ and the property $P(X)$. 
7: Probabilistic Model Checking

**Given:** Markov chain $\mathcal{M}$, LTL formula $\varphi$

**Goal:** compute the probability that $\varphi$ holds in $\mathcal{M}$

**Classic Approach:**
1. get NBA $\mathcal{B}$ for $\neg \varphi$
2. determinise $\mathcal{B} \leadsto$ DRA $\mathcal{A}$
3. analyse $\mathcal{M} \otimes \mathcal{A}$

**Problem:** determinisation of $\mathcal{B}$ is expensive

**Idea:** consider simpler constructions for determinisation

**Subset Construction:** fast, can yield an inconclusive answer

**Breakpoint Construction:** slower, might also be inconclusive

**Multi-Breakpoint Construction:** very slow, always conclusive
8: Monte Carlo Model Checking

- **Scalable** and applicable for large systems
- **Idea**: Instead of complete state space only consider parts
- **Randomly** sample paths
- If path is **counterexample**: property not satisfied
- Else: sample more paths
- **Result**: confidence that property is satisfied
Advanced Model Checking Techniques

9: Concurrent Model Checking Algorithms

- Tarjan’s algorithm used for finding strongly connected components (SCCs)
- Crucial in model checking
- DFS which tries to find backward edges to already visited nodes
- Idea: utilise multi-core processors
- Lift algorithm to concurrent algorithm
Analysis of Probabilistic Programs

Outline

Overview

Aims of this Seminar

Important Dates

Pointer and Shape Analysis

Advanced Model Checking Techniques

Analysis of Probabilistic Programs

Final Hints
Analysis of Probabilistic Programs

1: Sampling for Probabilistic Programs

- **Probabilistic programs** = ordinary programs + randomness

  ```
  x := 0 [0.5] 1;
  if(x=0) { x := x + (0 [0.5] 1); }
  observe(x > 0)
  ```

- **Inference**: What is the probability distribution of a program?
- **Sampling** = Inference through program execution
- **Problem**: Large number of samples needed
- **This paper**: Apply program analysis techniques prior to sampling to obtain more accepting samples
2: Slicing Probabilistic Programs

- A probabilistic program $P$ returns a distribution over return values
- Goal: Obtain a *simpler program* $\text{Slice}(P)$
  - Correctness: Slicing should preserve the distribution over return values
  - Efficiency: Slicing should be done as fast as possible
- Traditional program slicing techniques are *not* correct for probabilistic programs
- *This paper*: Correct and efficient approach for probabilistic program slicing
3: Sampling Functions for Probability Distributions

Shortcoming

Many programs generate only discrete probability distributions

This paper

- Presents a programming language that is expressive enough for
  - discrete probability distributions
  - continuous probability distributions
  - probability distributions that are neither
- Presents technique for formal reasoning about the language
- Uses examples from robotics:
  - Localization
  - People tracking
  - Mapping
4: Static Analysis of Probabilistic Programs

Problem
Approximate the probability that a program establishes a given assertion $\phi$.

Solution overview
Infer the whole program behaviour from finitely many executions:
- Choose finite set of executions with overall high probability
- Compute the probability of $\phi$ within this set of executions by *symbolic execution*
- Use this probability to give guaranteed bounds for the probability of $\phi$ in the whole program
- Instead of computing exact probabilities, approximate using *branch-and-bound techniques over polyhedra*

```plaintext
n := 0;
repeat
  n := n + 1;
  c := coin_flip(0.5);
until (c = heads);
return n
```
5: Probabilistic Termination

- Behaviour of *ordinary programs* entirely determined by input
  - Program either terminates or not
- Behaviour of *probabilistic programs* depends on randomness
  - Program terminates with some probability
- Probabilistic program terminates **almost-certainly** if it terminates with probability 1
- Proving almost-certain termination is extremely difficult (more difficult than halting problem)
- **In this paper:** a proof rule for proving almost-certain termination relatively easily for certain programs
Final Hints

Outline

Overview

Aims of this Seminar

Important Dates

Pointer and Shape Analysis

Advanced Model Checking Techniques

Analysis of Probabilistic Programs

Final Hints
### Final Hints

#### Some Final Hints

<table>
<thead>
<tr>
<th>Hints</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Take your time to <strong>understand</strong> your literature.</td>
</tr>
<tr>
<td>• Be <strong>proactive</strong>! Look for <strong>additional</strong> literature and information.</td>
</tr>
<tr>
<td>• Discuss the content of your report with other students.</td>
</tr>
<tr>
<td>• Be <strong>proactive</strong>! Contact your supervisor on <strong>time</strong>.</td>
</tr>
<tr>
<td>• Prepare the meeting(s) with your supervisor.</td>
</tr>
<tr>
<td>• Forget the idea that you can prepare a talk in a day or two.</td>
</tr>
</tbody>
</table>
Final Hints

Some Final Hints

Hints

- Take your time to understand your literature.
- Be proactive! Look for additional literature and information.
- Discuss the content of your report with other students.
- Be proactive! Contact your supervisor on time.
- Prepare the meeting(s) with your supervisor.
- Forget the idea that you can prepare a talk in a day or two.

We wish you success and look forward to an enjoyable and high-quality seminar!