Formal Verification Meets Machine Learning

Introduction

Summer Semester 2018; 12 April, 2018

Joost-Pieter Katoen et al.
Software Modeling and Verification Group
RWTH Aachen University

https://moves.rwth-aachen.de/teaching/ss-18/fvmml/

Software Modeling and Verification Chair | RWTH AACHEN UNIVERSITY

Outline

Overview

Aims of this Seminar

Important Dates

Verification of Properties

Learning of Invariants

Model Learning

Synthesis of Programs and Algorithms

Final Hints

Formal Verification Meets Machine Learning

Formal verification

- Computer-supported mathematical analysis methods for ensuring correctness of systems
  - model checking, theorem proving, ...
- Often applied to safety-critical systems
- Complementary to testing methods

Overview

Formal Verification Meets Machine Learning

Formal verification

- Computer-supported mathematical analysis methods for ensuring correctness of systems
  - model checking, theorem proving, ...
- Often applied to safety-critical systems
- Complementary to testing methods

Machine learning

- Algorithms learning from data that is observed in possibly unknown environments
  - autonomous systems, computer vision, video games, ...
- Increasingly applied in safety-critical settings

Formal Verification Meets Machine Learning

Formal verification

- Computer-supported mathematical analysis methods for ensuring correctness of systems
  - model checking, theorem proving, ...
- Often applied to safety-critical systems
- Complementary to testing methods

Machine learning

- Algorithms learning from data that is observed in possibly unknown environments
  - autonomous systems, computer vision, video games, ...
- Increasingly applied in safety-critical settings

Research issues

- Safety-related issues for machine learning
- Explainability in AI and in model checking
- Scalability and applicability of formal verification

Outline

Overview

Aims of this Seminar

Important Dates

Verification of Properties

Learning of Invariants

Model Learning

Synthesis of Programs and Algorithms

Final Hints

Aims of this Seminar

Goals

Aims of this seminar

- **Independent understanding** of a scientific topic
- Acquiring, reading and understanding **scientific literature**
- Writing of your **own report** on this topic
- **Oral presentation** of your results

Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

Requirements on Report

Your report

- Independent writing of a report of **10–15 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
  - $\geq$ 10 errors per page $\Longrightarrow$ abortion of correction
- **LaTeX template** will be made available on seminar web page

Aims of this Seminar

Requirements on Talk

Your talk

- Talk of **30 minutes**
- Available: projector, presenter, [laptop]
- Focus your talk on the **audience**
- **Descriptive** slides:
  - $\leq$ 15 lines of text
  - use (base) colors in a useful manner
  - number your slides
- **Language:** German or English
- No spelling mistakes please!
- Finish **in time**. Overtime is bad
- Ask for **questions**
- Have **backup slides** ready for expected questions
- **LaTeX template** will be made available on seminar web page

Outline

Overview

Aims of this Seminar

Important Dates

Verification of Properties

Learning of Invariants

Model Learning

Synthesis of Programs and Algorithms

Final Hints

Important Dates

Deadlines

- 14 May: Detailed outline of report due
- 11 June: Report due
- 2 July: Presentation slides due
- 19 July (?): Seminar

Important Dates

Deadlines

- 14 May: Detailed outline of report due
- 11 June: Report due
- 2 July: Presentation slides due
- 19 July (?): Seminar

Missing a deadline causes **immediate exclusion** from the seminar

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 (16 April)** via e-mail (`noll@cs.rwth-aachen.de`) or 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.

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 (16 April)** via e-mail (`noll@cs.rwth-aachen.de`) or 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.

Outline

Overview

Aims of this Seminar

Important Dates

Verification of Properties

Learning of Invariants

Model Learning

Synthesis of Programs and Algorithms

Final Hints

1: Safety Verification of Deep Neural Networks

Context

- Deep neural networks are impressive for image classification
- Minimal changes to the input image may yield a misclassification
- What does that mean in a safety-critical context?
- How to guarantee that deep neural networks are safe?

Topic

One of the first works on verifying neural networks:

- Verifying multi-layer neural networks based on Satisfiability Modulo Theory (SMT)
- Safety = invariance of a classification within a small neighbourhood of true image
- Verification works directly with the network code and can guarantee that adversarial examples, if they exist, are found

2: Verification of Markov Decision Processes Using Learning Algorithms

Context

- MDPs are transition systems with non-determinism and probabilities
- Their verification amounts to solve linear programs
- The size of these linear programs is the main bottleneck for MDP analysis

Topic

- Avoiding an exhaustive exploration of the state space by machine learning
- Alg 1: A heuristic-driven partial MDP exploration for reachability
- Alg 2: A sampling-based approach for reachability

Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

3: Smoothed Model Checking

Context

- CTMCs are transition systems with stochastic state delays
- These models are used in biology, performance analysis, etc.
- Verification assumes all transition rates to be known a priori

Topic

- For which parametric rate values does a CTMC satisfy a property?
- Predict the value of the satisfaction probability at all values of the uncertain parameters from individual model simulations at a finite (and generally rather small) number of distinct parameter values
- Exploits that satisfaction probabilities are smooth functions
- Approximates the satisfaction function by a sample from a Gaussian Process

4: Formal Verification of Neural Networks

Context

- Neural networks are layered systems mapping inputs to outputs
- The outcomes of neural networks come with weak guarantees
- Can formal verification be applied to prove hard guarantees?

Topic

- Verifying feed-forward neural networks where all nodes are piece-wise linear functions
- Adds a global linear approximation of the overall network behavior
- Exploits this using SAT-inspired algorithms such as unit propagation
- Infers additional conflict clauses from the analysis results during the search

5: Verification and Control of Partially Observable Probabilistic Systems

Context

- Partially observable systems are (probabilistic) transition systems in which only observations are identifiable; states are not
- How to steer a robot when it does not recognize its environment completely?

Topic

- Automated verification techniques for partially observable, probabilistic systems
- Considers extension of MDPs and of timed automata
- Synthesize a controller for POMDPs which makes a given property true
- Exploits a grid-based abstraction of the uncountable belief space induced by partial observability
- Applications: task and network scheduling, computer security and planning

6: Counterexample Explanation for Probabilistic Systems

Context

- Model checking yields counterexamples if model refutes property
- For transition systems, counterexamples are finite paths
- For probabilistic systems, they are (possibly infinite) sets of finite paths
- How to obtain such counterexamples, and how to represent them succinctly?

Topic

- Use decision trees to compactly represent counterexamples

Outline

Overview

Aims of this Seminar

Important Dates

Verification of Properties

Learning of Invariants

Model Learning

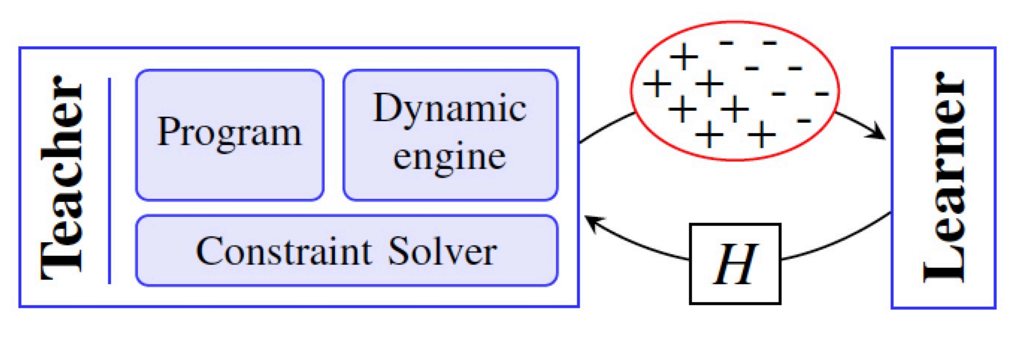
Synthesis of Programs and Algorithms

Final Hints

7: Learning Loop Invariants

Main principle

2 parts: a Learner and a Teacher. In each iteration of the algorithm,
a learner generates a candidate invariant, and the teacher validates the conjecture and,
if necessary, generates feedback in a form of a positive or negative example.
The learner uses the example to improve its conjecture.
Termination once the teacher determines that an inductive invariant is found.



7: Learning Loop Invariants

Context

- Finding loop invariants is the main obstacle in program verification
- It is this task that makes program verification undecidable
- Can learning be used to automatically synthesize loop invariants?
- Use that invariants represent over-approximations of the set of reachable program loop states

Topic

- Represents loop invariants by a kind of finite-state automata
- Synthesizes invariants by learning using examples, counter-examples, and implications
- Type 1: Learns Boolean combinations of numerical invariants for scalar variables
- Type 2: Learns quantified invariants for arrays and dynamic lists

Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

8: Learning Data Structure Invariants

Context

- Can learning be used to obtain loop invariants over linear data structures?

Topic

- Represents (quantified) invariants over linear data structures by quantified data automata over words
- Poly-time active learning algorithms for such automata
- Every QDA has a unique minimally-over-approximating “elastic” QDA
- Elastic QDAs can be represented by decidable logics
- Learning such automata from samples obtained by running programs

9: Syntax-Guided Invariant Synthesis

Context

- SyGiS problem = find invariant generated by a given grammar that meets a specification
- Idea: use correctness specification + a syntactic template for the desired invariant
- Approach:
  1. Construct a formal grammar based on the symbolic program encoding
  2. Use probabilistic search through the candidate formulas in that grammar

Topic

Accelerating SyGiS by:

- enumerative learning with inductive- subset extraction
- leverages counterexamples-to-induction and
- Craig interpolation-based bounded proofs.

Experiments show that this leads to significant improvements

10: Synthesizing Inductive Invariants

Context

- IC3 is a popular SAT-MC technique in hardware and software verification for safety properties

  iteratively conjecture safe candidate invariants until either a candidate is proven inductive or a counterexample is found
- ICE is a rather new ML technique for learning loop invariants (see topic 7)

Topic

- Goal: study the similarities and differences between the two frameworks
- RICE = ICE + relative induction
- Show how IC3 can be implemented as an instance of RICE.

Outline

Overview

Aims of this Seminar

Important Dates

Verification of Properties

Learning of Invariants

Model Learning

Synthesis of Programs and Algorithms

Final Hints

11: Generating Models of Communication Protocols

Context

- Automata learning learns an automaton by examples (finite words)
- Recent progress learns register automata = automata + data
- Using abstraction techniques, these algorithms can be applied to complex systems
- Model learning emerges as an effective bug-finding technique
- Applications include banking cards, network protocols, and legacy software

Topic

- A framework which adapts regular inference to include data parameters in messages and states for generating components with large or infinite message alphabets
- Examples: regular inference of session initiation protocol (SIP) and TCP

12: Learning Finite Automata

Context

- Automata learning focuses on learning deterministic finite-state automata
- For DFA, minimal automata are canonical
- But NFAs can be exponentially more succinct; can we learn NFA?

Topic

- How can automata learning be adapted to obtain small NFA?
- What is a canonical representation of “minimal” NFA?

13: Learning and Planning with Timing Information in MDPs

Context:

- MDPs are probabilistic automata used for decision making and planning
- Main aim: determine an optimal policy for the decisions given a certain objective
- Typical assumption: actions are instantaneous

Topic

- Learn the duration of courses of action that an agent might take
- A spectral algorithm for learning option duration models
- Application: navigation of robots

Synthesis of Programs and Algorithms

Outline

Overview

Aims of this Seminar

Important Dates

Verification of Properties

Learning of Invariants

Model Learning

Synthesis of Programs and Algorithms

Final Hints

14: Policy Learning in Continuous-Time Markov Decision Processes

Context:

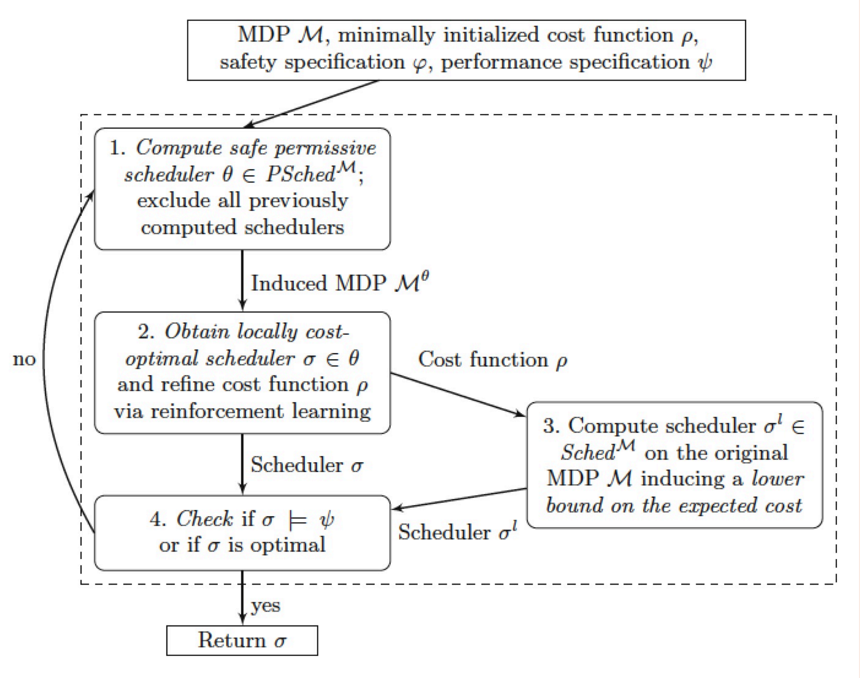
- Continuous-time MDPs extend MDPs with random state delays
- Applications: financial mathematics, control of populations, scheduling, . . .
- Aim: compute an optimal policy to control the CTMDP to maximise the probability of satisfying an objective
- Statistical model checking (SMC) = discrete-event simulation to check temporal logic objectives

Topic

- New method to obtain optimal policies for CTMDPs
- Key: use a stochastic moment-based gradient ascent algorithm to guide the search for optimal policies
- Gaining performance on an epidemiology model and a queuing system

15: Safety-Constrained Reinforcement Learning for Markov Decision Processes

Topic



16: Learning Static Analyzers

Context

- Static program analysis is the analysis of computer software that is performed without actually executing programs
- The Clang Static Analyzer is a source code analysis tool that finds bugs in C
- Popular approach: use inference rules to steer the static analysis
- How to obtain the right inference rules?

Topic

- A new, automated approach for creating static analyzers
- Key idea: learn inference rules from a dataset of programs
- Two ingredients:
  1. a synthesis algorithm for learning a candidate analyzer from a given dataset
  2. a counter-example guided learning procedure to generate programs outside the dataset
- Case study: learn JavaScript static analysis rules for points-to analysis

17: Learning Disjunctions of Predicates

Topic

- Let F be a set of predicates (i.e., boolean functions)
- Goal: learn the class $F_\vee$ of any disjunction of predicates in F.
- Learning algorithm SPEX, which learns any function in $F_\vee$ with polynomially many queries.
- Imposing some computational complexity conditions on the set of predicates, SPEX runs in polynomial time.
- Two (practical) cases: polytopes, and conjunctions of form $x_i > x_j$
- Integration of this technique in programming by example (PBE)

18: Learning Explanatory Rules from Noisy Data

Context

- ILP (Inductive Logic Programming):
  - Given a set of positive examples, and a set of negative examples, ILP constructs a logic program that entails all positive examples but does not entail any negative example
- Major shortcoming: inability to handle noisy, erroneous, or ambiguous data

Topic

- Extend ILP with the possibility to use neural networks for learning programs
- The δILP achieves reasonable performance with up to 20% of mislabelled training data
- It is able to solve moderately complex tasks requiring recursion and predicate invention
- Examples: length of a list, (grand)parent in a tree, cyclicity of graphs

19: Discovery of Divide-&-Conquer Algorithms

Context

- The divide-and-conquer (D-and-C) paradigm is popular in algorithm design
- Examples include many sorting algorithms, dynamic programming (DP)
- Key for DP problems is to find a recurrence relation
- Is it possible to automate the generation of a (parallel) program from such recurrences? In a memory-efficient manner?

Topic

- An algorithm that for DP problems automatically discovers (parallel) recursive D-and-C algorithms from iterative descriptions of DP recurrences
- Several synthesized algorithms significantly outperform existing parallel algorithms

20: Multi-Objective Policy Generation for Mobile Robots

Context

- Multi-objective model checking for Markov decision processes
- Probabilistic time-bounded guarantees on successful task completion, whilst also trying to satisfy soft goals

Topic

- Setting: a stochastic model of the robots environment and action execution times, a set of soft goals, and a formal task specification in co-safe LTL
- Contribution: new technique to do multi-objective model checking on MDPs
- Case: generate policies on a delivery task in a care home scenario, where the robot also tries to engage in entertainment activities with the patients

Outline

Overview

Aims of this Seminar

Important Dates

Verification of Properties

Learning of Invariants

Model Learning

Synthesis of Programs and Algorithms

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.

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!