Seminar *Trends in Computer-Aided Verification*

Introduction

Winter 2024/25; October 9, 2024

Thomas Noll et al.
Software Modeling and Verification Group
RWTH Aachen University

https://moves.rwth-aachen.de/teaching/ws-2024-25/cav/

Software Modeling and Verification Chair | RWTH AACHEN UNIVERSITY

Outline

Formal Verification Methods

Formal verification methods

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

Formal Verification Methods

Formal verification methods

- Rigorous, mathematically based techniques for the specification, development 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
  - neural network, Markov chain, source code, ...
- According to underlying mathematical theories
  - model checking, theorem proving, static analysis, ...

Outline

Goals

Aims of this seminar

- Independent understanding of a scientific topic
- Acquiring, reading and understanding scientific literature
  - given references sufficient in most cases
- Writing of your own report on this topic
  - far more that just a translation/rewording
  - usually an “extended subset” of original literature
    - “subset”: present core ideas and omit too specific details (e.g., related work or optimisations)
    - “extended”: more extensive explanations, examples, ...
    - discuss contents with supervisor!
- Oral presentation of your results
  - can be “proper subset” of report
  - generally: less (detailed) definitions/proofs and more examples

Requirements on Report

Your report

- Independent writing of a report of 12–15 pages
- First milestone: detailed outline
  - not: “1. Introduction/2. Main part/3. Conclusions”
  - rather: overview of structure (section headers, main definitions/theorems) and initial part of main section (one page)
- 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
  - LaTeX template will be made available on seminar web page
- Language: German or English
- We expect the correct usage of spelling and grammar
  - $\geq$ 10 errors per page $\Longrightarrow$ abortion of correction

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
  - LaTeX/beamer template will be made available on seminar web page
- Language: German or English
- No spelling mistakes please!
- Finish in time. Overtime is bad
- Ask for questions
- Have backup slides ready for expected questions

Outline

Important Dates

Deadlines

- October 11: Topic preferences due
- November 11: Detailed outline due
- December 9: Full report due
- January 13: Presentation slides due
- February 3–5 (?): Seminar talks

Important

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 here or via e-mail (`noll@cs.rwth-aachen.de`) by Friday (October 11).
- 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 early next week.
- Then also your supervisor will be indicated.

Withdrawal

- You have up to one week (!) to refrain from participating in this seminar (after topic assignment).
- 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

A. Verification of Neural Networks [Christopher Brix]

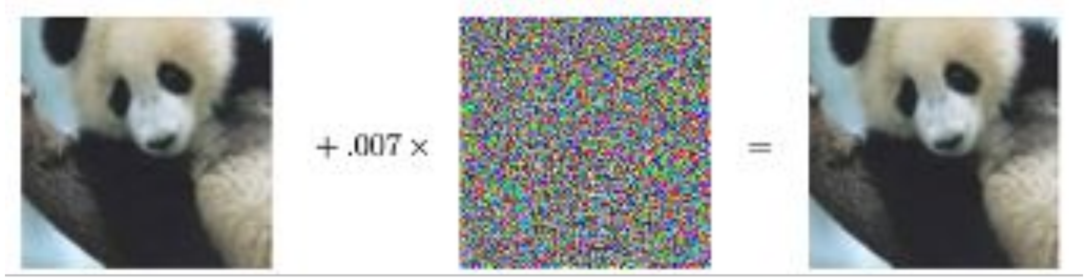
B. Compositional Verification of Probabilistic Systems [Hannah Mertens]

C. Analysis of Partially Observable Stochastic Systems [Alexander Bork]

D. Static Analysis of Quantum Programs [Thomas Noll]

Final Hints

Motivation



- Verification guarantees robustness to perturbations
  - Formal process, sound bounds on network behaviour
- Novelty Detection identifies unexpected inputs
  - Heuristic approach
  - Aims to avoid “guessing” for inputs the network was not trained on

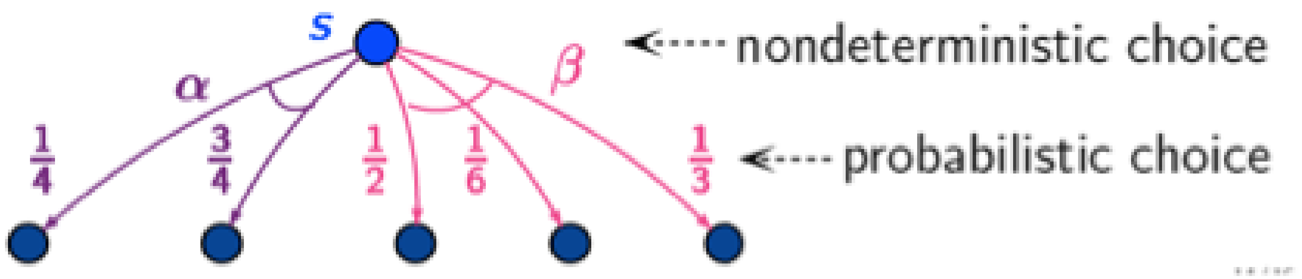
Verification of Neural Networks

1. Abstraction-Based Verification with Intervals and Zonotopes
   – Introduction into NN verification
   – More formal
   – Network behaviour needs to be approximated
   – Aws Albarghouthi: *Introduction to Neural Network Verification*, textbook, pp. 83–108
2. Shared Certificates for Neural Network Verification
   – The verification of one (robustness) property can be reused to help proving another one
   – Demonstrates that different input perturbations require similar proofs
   – Marc Fischer, Christian Sprecher, Dimitar I. Dimitrov, Gagandeep Singh, Martin Vechev: *Shared Certificates for Neural Network Verification*, CAV 2022
3. Detecting Novel Inputs
   – Networks guess: after training on animals, it may return “cat” for cars
   – Problem: Identify inputs that are outside the training domain (“don’t know”)
   – Computes clusters for known inputs, input outside those clusters are considered out-of-distribution
   – Thomas A. Henzinger, Anna Lukina, Christian Schilling: *Outside the Box: Abstraction-Based Monitoring of Neural Networks*, ECAI 2020

Outline

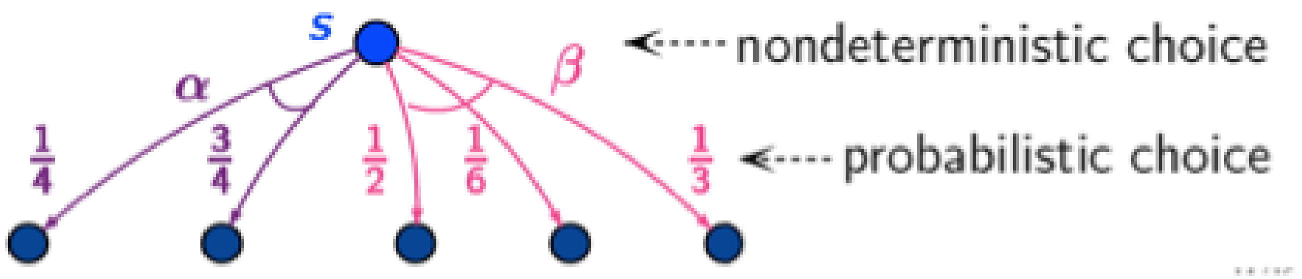
Probabilistic Systems:

e.g., Markov decision processes (MDPs)

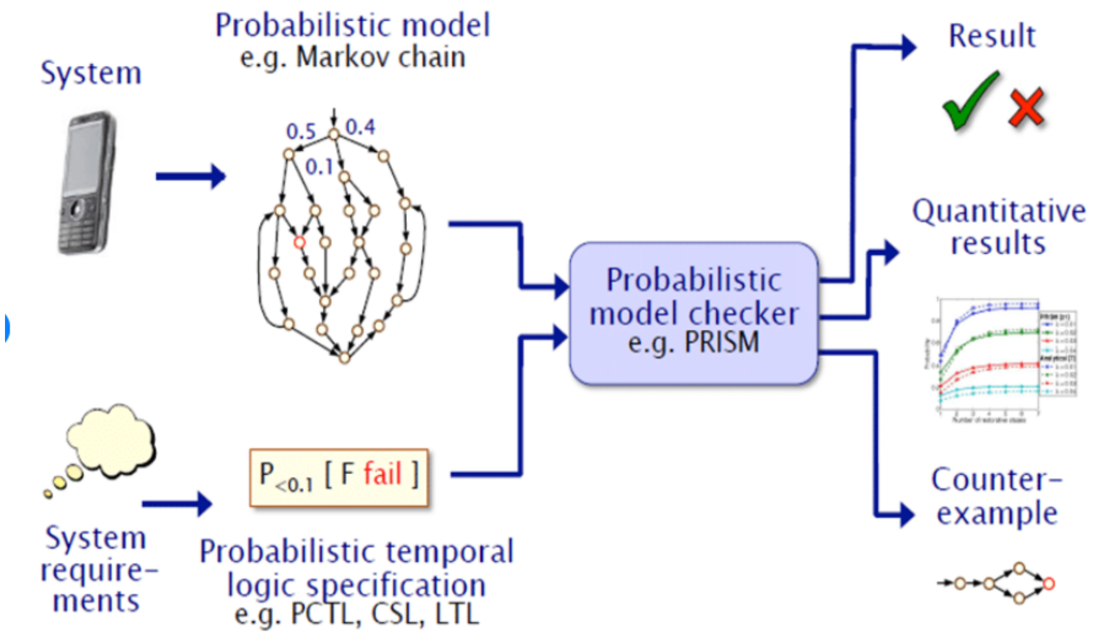


Verification of Probabilistic Systems

Probabilistic Systems:
e.g., Markov decision processes (MDPs)

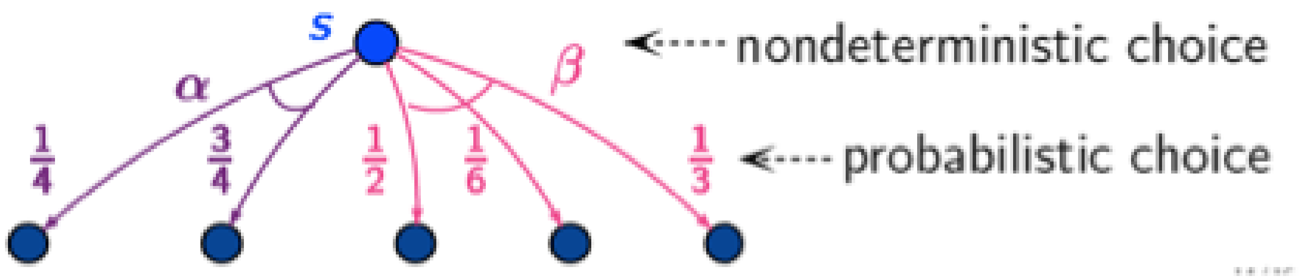


Verification:

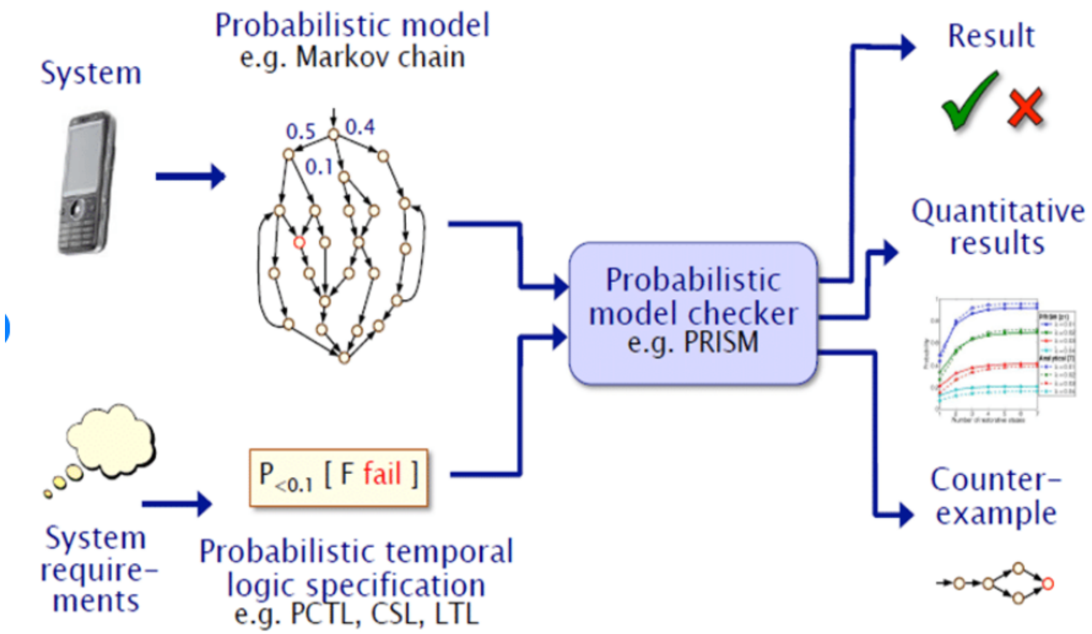


Probabilistic Systems:

e.g., Markov decision processes (MDPs)



Verification:



Compositional Verification:

- Reduce peak memory consumption by reasoning about individual parts and putting results together
- Exploit the existence of isomorphic parts of the state space

Assume-Guarantee Reasoning

Framework for analysing parallel composition of communicating programs:

- Communicating programs: infinite-state C-like programs that can synchronously read and write messages over communication channels
- Composition formalism: Assume-Guarantee-Repair (AGR)
- AGR verifies that a program satisfies a set of properties and repairs the program if the verification fails
- Employs Assume-Guarantee (AG) rules: e.g.,

Rule ASYM-AG

$$\frac{\begin{array}{ll}\text{(Premise 1)} & \langle A\rangle M_1 \langle P\rangle \\ \text{(Premise 2)} & \langle true\rangle M_2 \langle A\rangle\end{array}}{M_1 \parallel M_2 \models P}$$

  "If M_1 under assumption A satisfies property P and any system containing M_2 as a component satisfies A, then the parallel composition $M_1 \parallel M2$ satisfies P."
- Hadar Frenkel, Orna Grumberg, Corina S. Păsăreanu, Sarai Sheinvald: *Assume, guarantee or repair: a regular framework for non regular properties*, STTT 2022

Compositional Strategy Synthesis

Framework for strategy synthesis in parallel composition of stochastic games:

- Stochastic two-player game: two types of nondeterminism
  - Player $\Box$ (uncontrollable environment)
  - Player $\Diamond$ (controllable part)
- Compose a winning strategy for $\Diamond$ in the composed system $G_1 \parallel G_2 \parallel \ldots$ out of strategies in the individual components $G_1, G_2, \ldots$ via assume-guarantee (AG) rules
- N. Basset, M. Kwiatkowska, C. Wiltsche: *Compositional strategy synthesis for stochastic games with multiple objectives*, Information and Computation 2018

Circular Assume-Guarantee Reasoning

Algorithm for circular AG reasoning of transition systems:

- Previous work: automation restricted to acyclic AG rules
- Employ a circular AG rule and automate the application of the rule CIRC-AG by automatically building the assumptions g_1, g_2

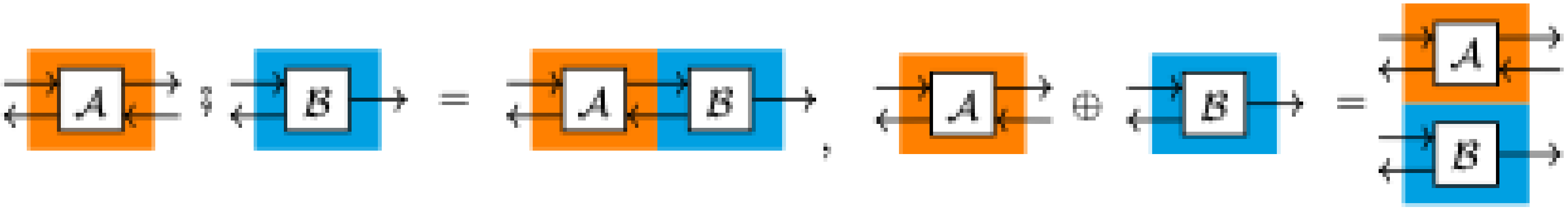
$$\begin{array}{ll} \text{(Premise 1)} & M_1 \models g_2 \rhd g_1 \\ \text{(Premise 2)} & M_2 \models g_1 \rhd g_2 \\ \text{(Premise 3)} & g_1 \mid\mid g_2 \models P \\ \hline & M_1 \mid\mid M_2 \models P \end{array}$$

- Karam Abd Elkader, Orna Grumberg, Corina S. Păsăreanu, Sharon Shoham: *Automated circular assume-guarantee reasoning*, Formal Aspects of Computing 2018

Compositional Model Checking

Framework for analysing sequentially composed MDPs:

- Composition formalism: string diagrams
- String diagrams of MDPs are MDPs composed by algebraic operations:



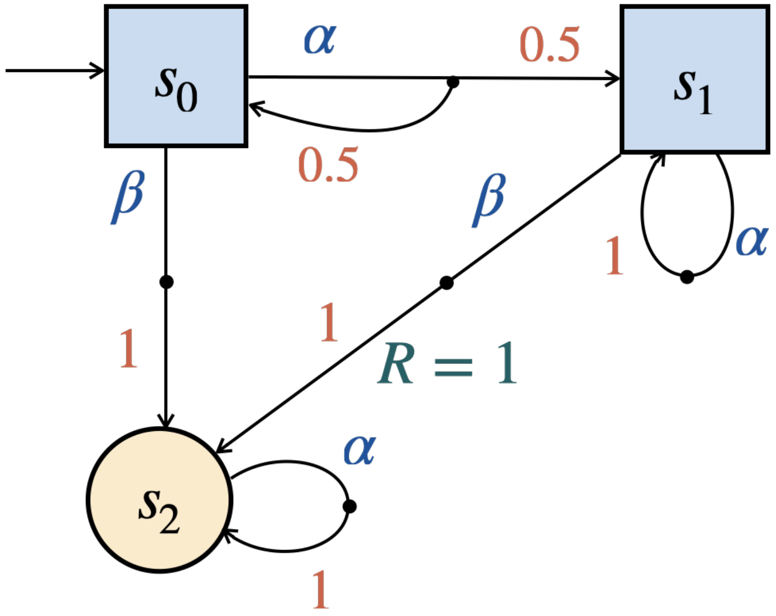
- Consider the schedulers in a subMDP which form a Pareto curve on a combination of local objectives.
- Employ multi-objective model checking of MDPs to obtain a novel compositional algorithm for MDPs compositionally defined by string diagrams.
- Kazuki Watanabe, Marck van der Vegt, Ichiro Hasuo, Jurriaan Rot, Sebastian Junges: *Pareto Curves for Compositionally Model Checking String Diagrams of MDPs*, TACAS 2024

Outline

Efficient Computation of Belief Values

Spaan, Vlassis: *Perseus: Randomized Point-based Value Iteration for POMDPs.* JAIR 24 (2005)

- Partially Observable MDPs (POMDPs): modeling formalism for planning in AI
  - non-deterministic choice & probabilistic branching
  - partially observable states
- Main question: what choices maximise expected rewards?
- Point-based value iteration methods are effective approximation techniques
- *Perseus* uses randomisation for speeding up computations



Planning under Constraints

Poupart et al.: *Approximate Linear Programming for Constrained Partially Observable Markov Decision Processes.* AAAI 2015

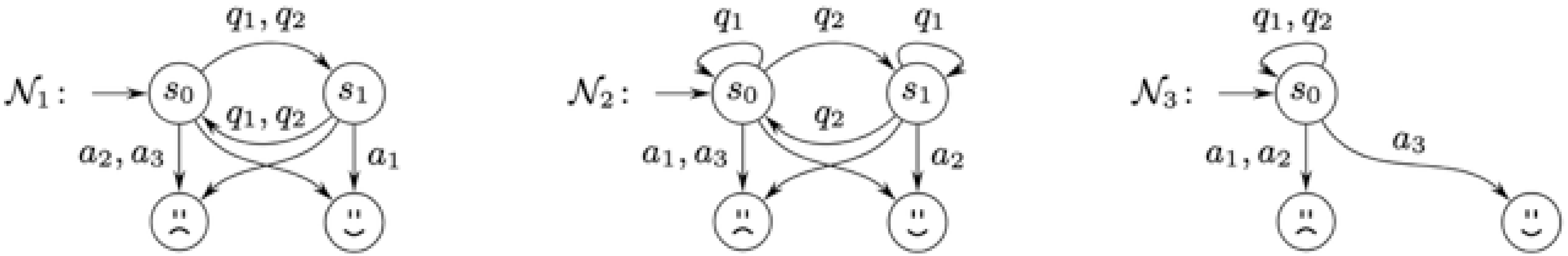
- Constrained POMDPs: POMDPs with constraints on the expected costs
- Exact solution methods often complex
- Use linear programming to approximate the solution

$$\text{maximise } E\left[\sum_t \gamma^t R(s_t, a_t)\right]$$

$$\text{subject to } E\left[\sum_t \gamma^t C_k(s_t, a_t)\right] \leq c_k \qquad \forall k$$

Multi-Environment Models

van der Vegt, Jansen, Junges: *Robust Almost-Sure Reachability in Multi-Environment MDPs.* TACAS 2023



- MEMDP: models different environments over the same state space
- Exact environment is unknown
- Examples: guessing a password, navigating with unknown obstacle positions, . . .
- Objective: find one strategy that almost-surely reaches a target in all environments
- Strongly related to POMDP problems

Outline

Motivation

Static (Program) Analysis

Static analysis is a general method for automated reasoning on artefacts such as requirements, design models, and programs.

Distinguishing features

Static: based on source code, not on (dynamic) execution

- in contrast to testing, profiling, or run-time verification

Automated: “push-button” technology, i.e., little user intervention

- in contrast to interactive “theorem-proving” approaches

(Main) Applications

- Initially (since 1970s): compiler optimisations and synthesis of efficient code
- Now: support for all phases of software development
  - verification of specifications
  - verification of program correctness
  - certification of critical software
  - refactoring and maintenance of applications, ...

Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

Detecting Bugs

- Pengzhan Zhao, Xiongfei Wu, Zhuo Li, Jianjun Zhao: *QChecker: Detecting Bugs in Quantum Programs via Static Analysis*, Q-SE 2023
- Introduces static analysis tool QChecker that supports finding bugs in quantum programs in Qiskit
- Two main modules:
  - extracting program information based on abstract syntax tree (AST)
  - detecting bugs based on patterns
- Patterns derived from real quantum bugs in previous studies
  - Incorrect uses of quantum gates, Measurement related issues, Incorrect initial state, ...

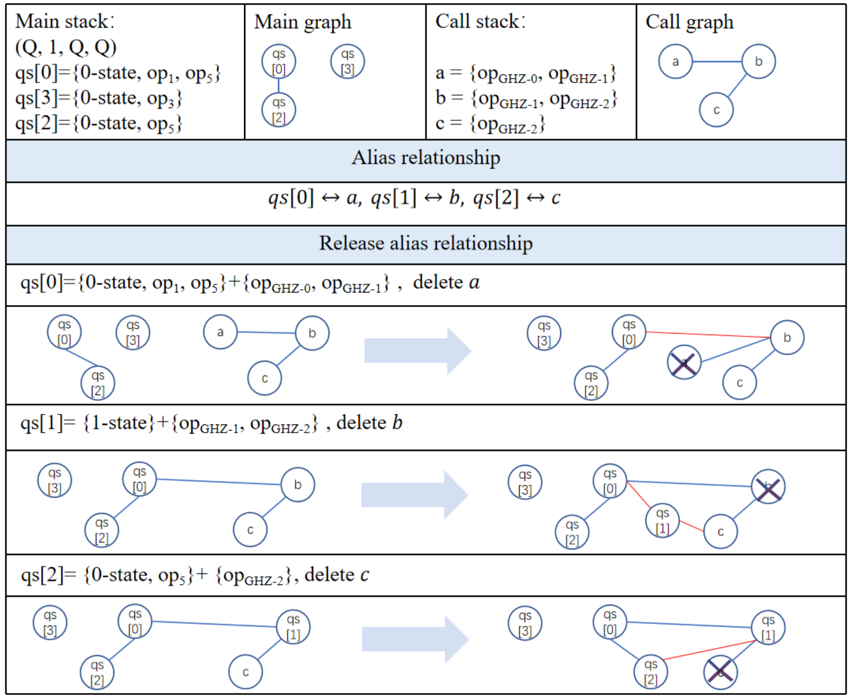
```
simulator = Aer.get_backend("qasm_simulator")

qreg = QuantumRegister(3)
creg = ClassicalRegister(3)
circuit = QuantumCircuit(qreg, creg)

circuit.h(0)
circuit.h(2)
circuit.cx(0, 1)
circuit.measure([0,1,2], [0,1,2])
job = execute(circuit, simulator, shots=1000)
result = job.result()
counts = result.get_counts(circuit)
print(counts)
```

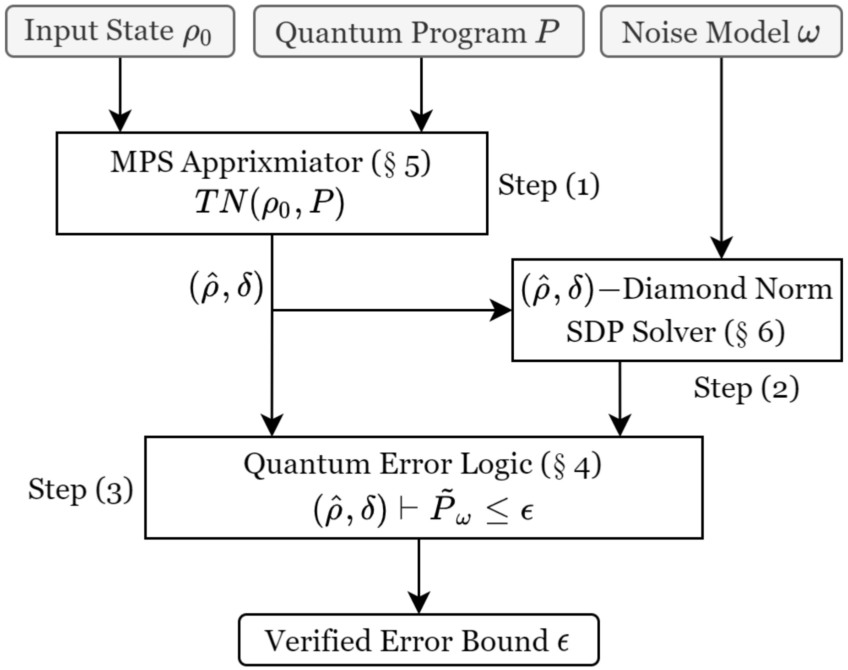
Entanglement Analysis

- Shangzhou Xia, Jianjun Zhao: *Static Entanglement Analysis of Quantum Programs*, Q-SE 2023
- Entanglement causes qubits to become mutually dependent
- Plays a crucial role in quantum computation
- Performing measurements requires considering the entanglement information
- Here: first static entanglement analysis method for quantum programs in Q#



Error Analysis

- Runzhou Tao, Yunong Shi, Jianan Yao, John Hui, Frederic T. Chong, Ronghui Gu: *Gleipnir: Toward Practical Error Analysis for Quantum Programs*, PLDI 2021
- Error analysis is essential for the design, optimization, and evaluation of Noisy Intermediate-Scale Quantum (NISQ) computing
- Here: novel methodology toward practically computing verified error bounds
- Can be used to evaluate the error mitigation performance of quantum compiler transformations
- Suitable for real-world quantum programs with 10 to 100 qubits



The LintQ Static Analysis Framework

```python
qc = QuantumCircuit(2, 2)
qc.h(1)
qc.cx(1, 0)
qc.measure(0, 0)
qc.measure(1, 1)
qc.z(0) # Problem: Qubit 0 has collapsed
qc.measure(0, 0)
```

```
from Measurement m, Gate g, int q
where
  mayFollowDirectly(m, g, q)
  and not g.isConditional()
select gate, "Gate after measurement
     on qubit " + q
```

- Matteo Paltenghi, Michael Pradel: *Analyzing Quantum Programs with LintQ: A Static Analysis Framework for Qiskit*, FSE 2024
- Uses abstractions for reasoning about common concepts in quantum computing (without referring to details of underlying quantum computing platform)
- Offers an extensible set of ten analyses that detect likely bugs
  - operating on corrupted quantum states, redundant measurements, incorrect compositions of sub-circuits, ...

Outline

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!