Seminar *Trends in Computer-Aided Verification*

Introduction

Winter 2025/26; October 20, 2025

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

`https://moves.rwth-aachen.de/teaching/ws-2025-26/cav/`

Chair of
Software Modeling
and Verification
(Computer Science 2)
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

Chair of Software Modeling and Verification (Computer Science 2)
RWTH AACHEN UNIVERSITY

Outline

Important Dates

Deadlines

- October 24: Topic preferences due
- November 24: Detailed outline due
- December 15: Full report due
- January 12: Presentation slides due
- February 2–3 (?): 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 24).
- 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.

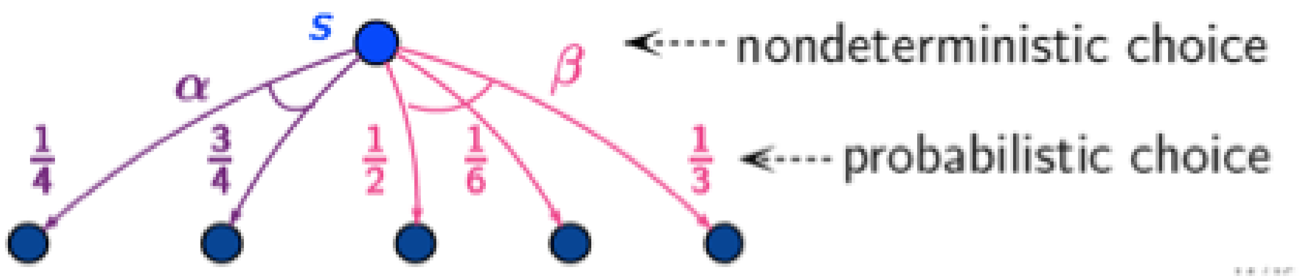
Withdrawal

- 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

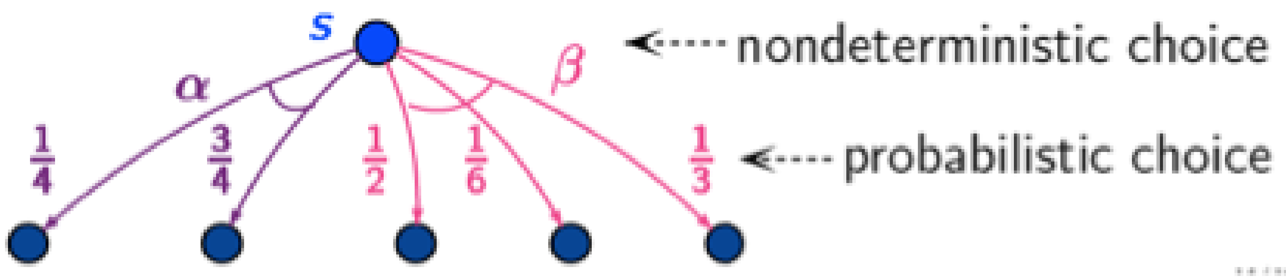
Probabilistic Systems:

e.g., Markov decision processes (MDPs)

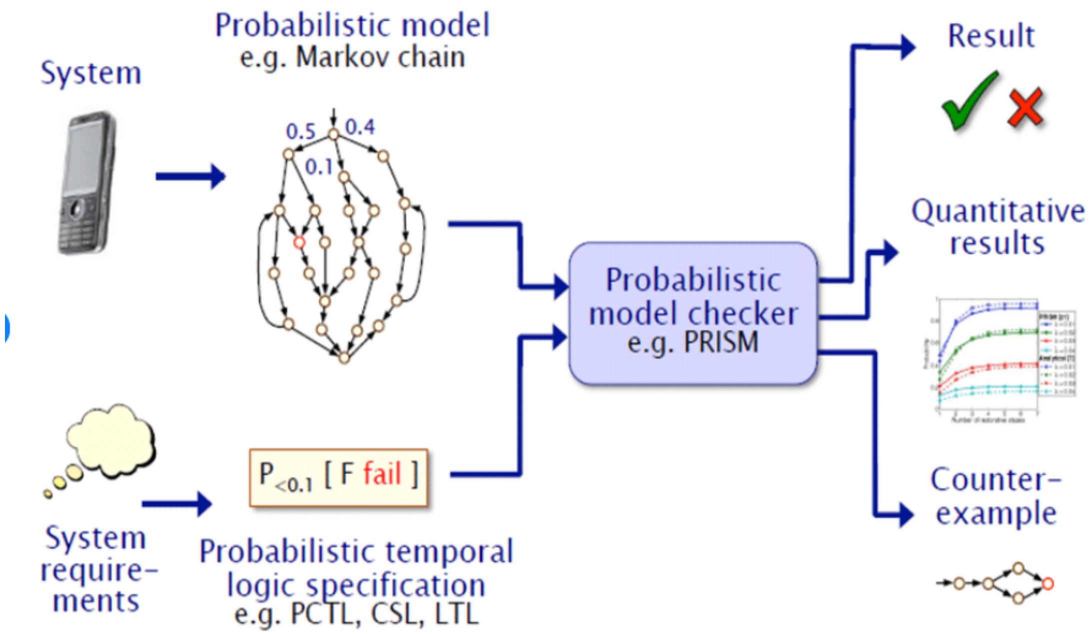


Probabilistic Systems:

e.g., Markov decision processes (MDPs)

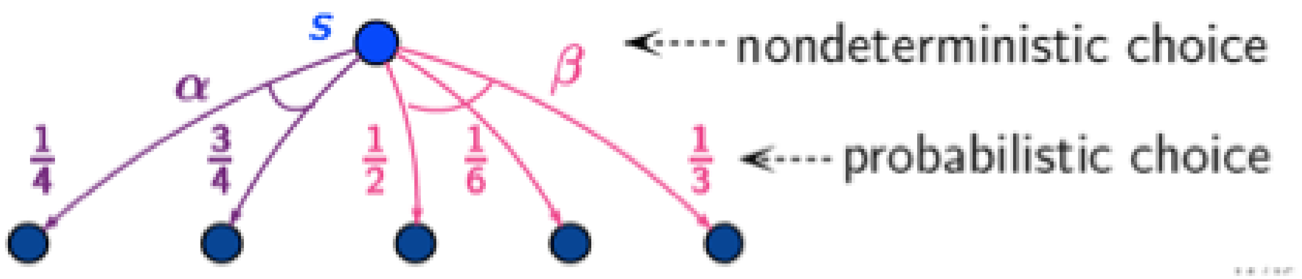


Verification:

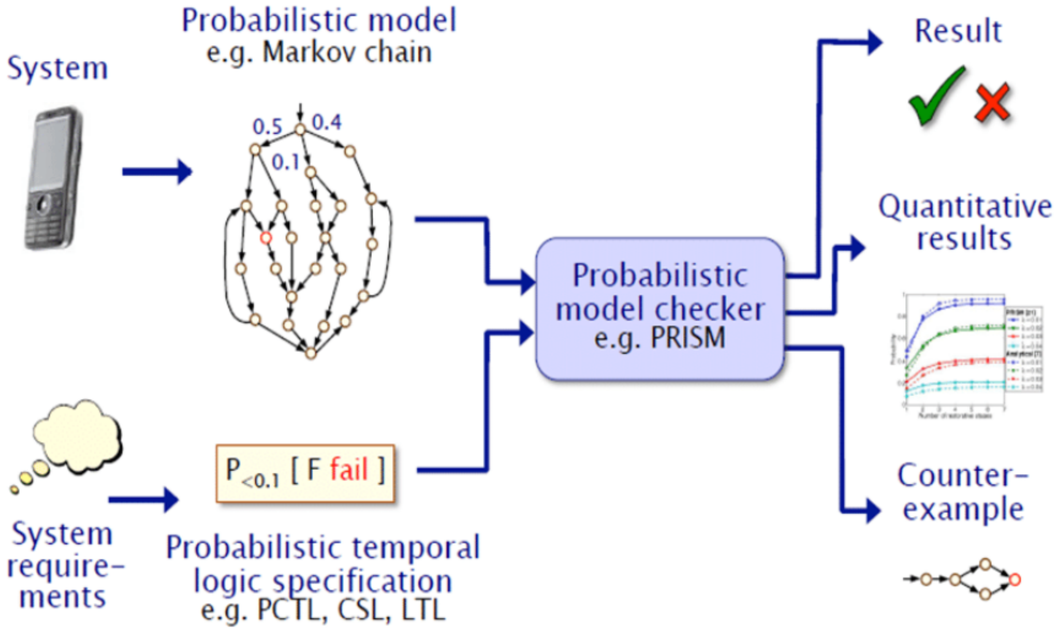


Probabilistic Systems:

e.g., Markov decision processes (MDPs)



Verification:



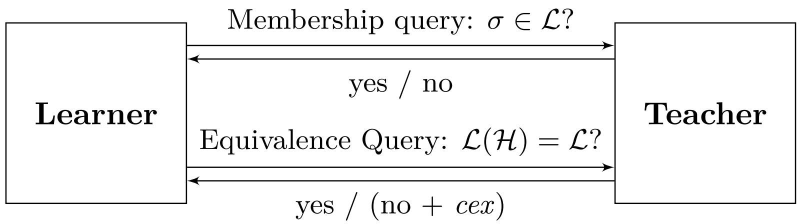
Compositional Verification:

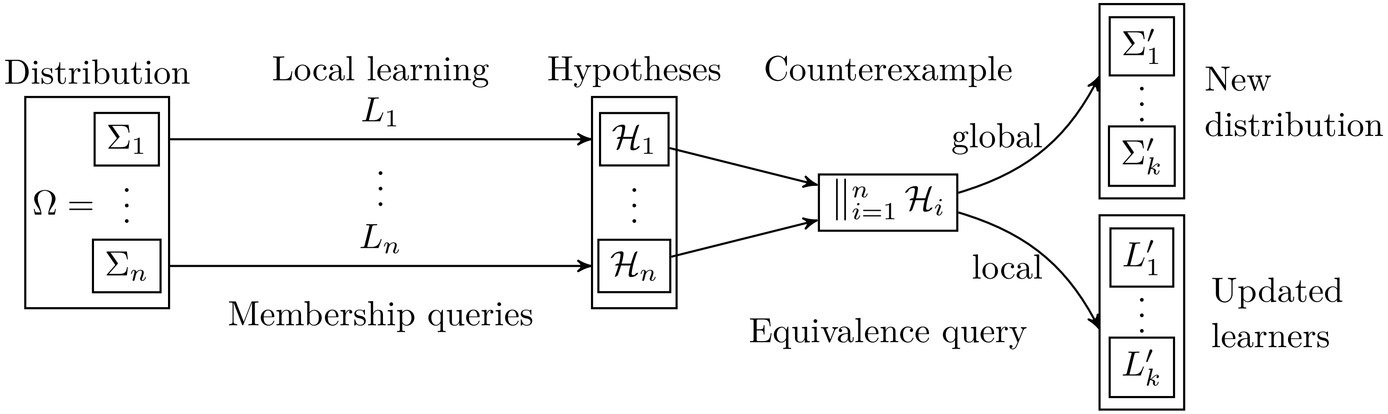
- Reduce peak memory consumption by reasoning about individual parts and putting results together
- Exploit the existence of isomorphic parts of the state space

1. Compositional Learning

Algorithm for compositional learning of automata by alphabet refinement:

- Automata learning: infers automata models of systems from behavioural observations
- Current trend: compositional approaches for concurrent systems
- Approach: automatic refinement of global alphabet into component alphabets while learning the component models



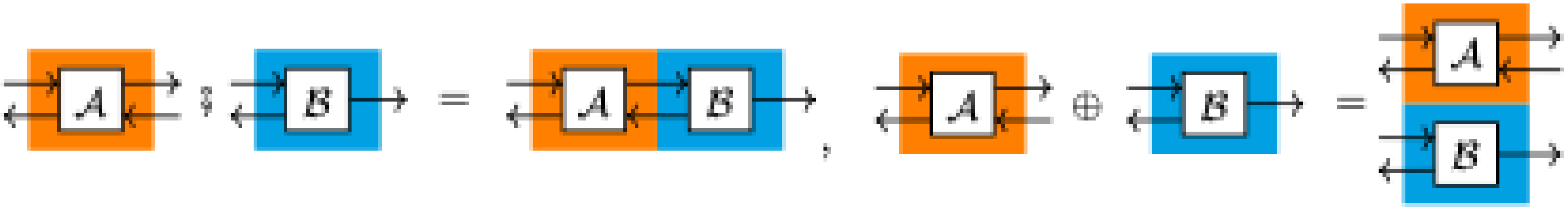


- Léo Henry, Mohammad Reza Mousavi, Thomas Neele, Matteo Sammartino: *Compositional Active Learning of Synchronizing Systems Through Automated Alphabet Refinement*, CONCUR 2025

2. 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 sub-MDP 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

3. Assume-Guarantee Reasoning

Framework for analysing systems with two parallel components:

- One Markov Decision Process (MDP) as controller model
- One Partially Observable MDP (POMDP) as environment model
- Verification employing Assume-Guarantee (AG) rules: e.g.,

$$\frac{\begin{array}{c} 1 : \mathcal{L}_1 || \mathcal{A} \models \psi \\ 2 : \mathcal{L}_2 \preceq^+ \mathcal{A} \end{array}}{\mathcal{L}_1 || \mathcal{L}_2 \models \psi}$$

  "If $\mathcal{L}_1$ under assumption *A* satisfies property ψ and any system containing $\mathcal{L}_2$ as a component satisfies *A*, then the parallel composition $\mathcal{L}_1 \parallel \mathcal{L}_2$ satisfies ψ."
- Xiaobin Zhang, Bo Wu, Hai Lin: *Assume-guarantee reasoning framework for MDP-POMDP*, CDC 2016

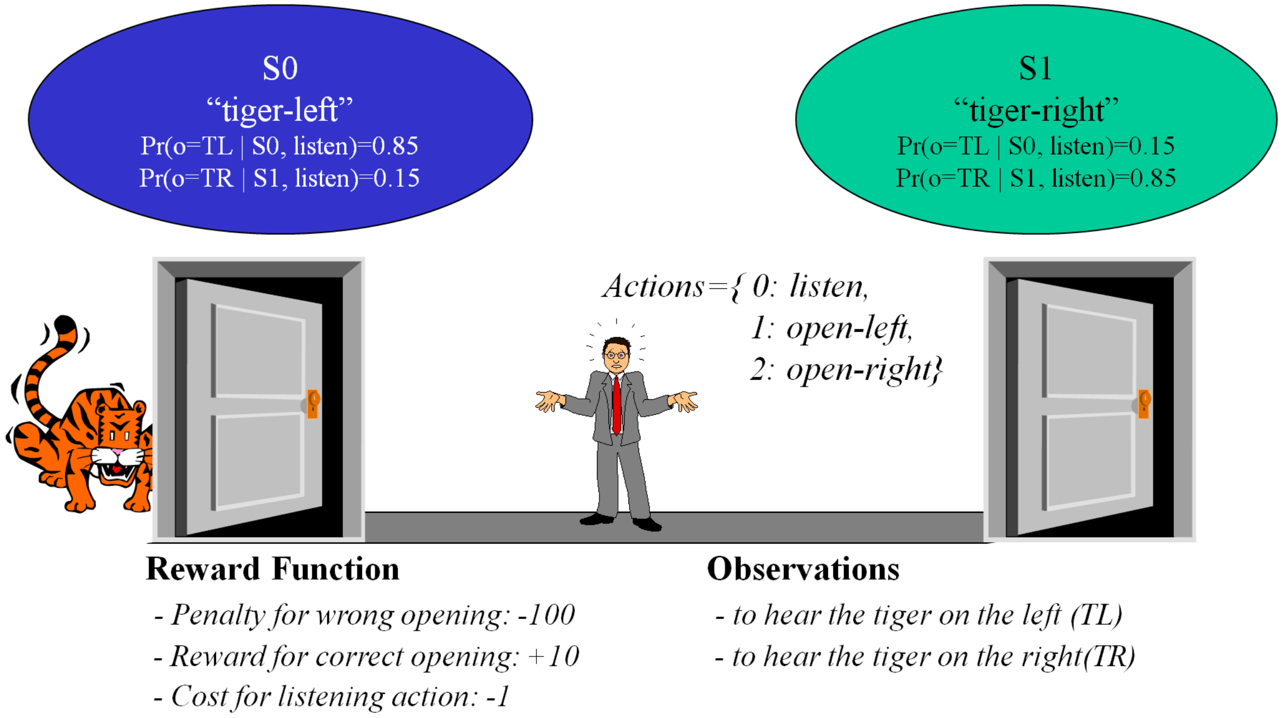
4. Compositional Strategy Synthesis

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

Outline

Partially Observable MDPs (POMDPs): modeling formalism for planning in AI

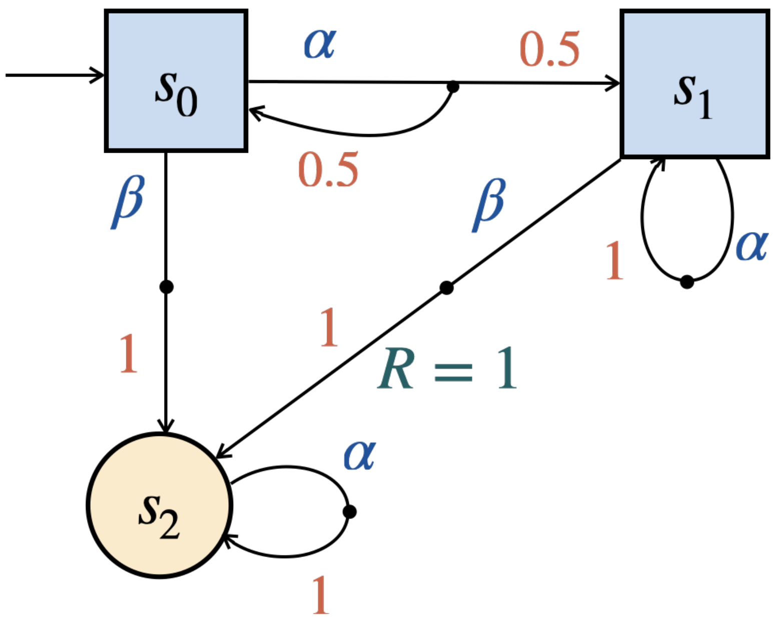


- non-deterministic choice & probabilistic branching
- partially observable states
- agents’ (partial) knowledge represented by belief state

1. 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



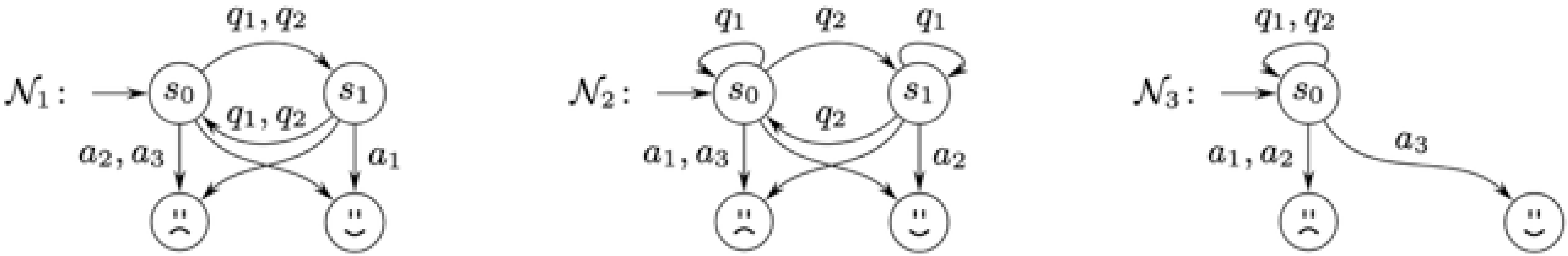
Poupart et al.: *Approximate Linear Programming for Constrained Partially Observable Markov Decision Processes.* AAAI 2015

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

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

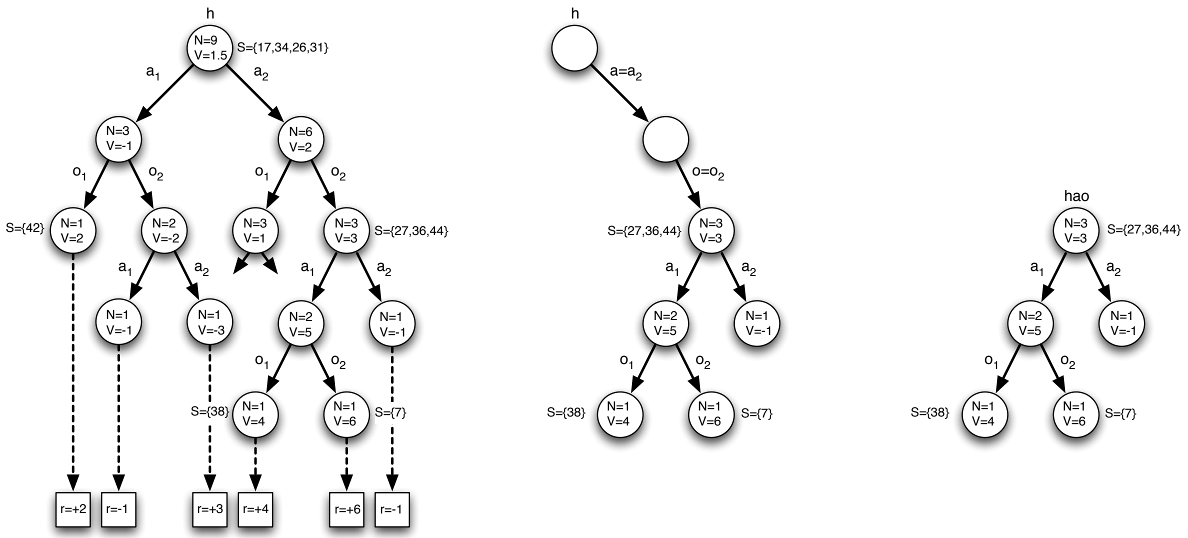
$$\text{subject to } E\left[\sum_t \gamma^t C_k(s_t, a_t)\right] \leq c_k \qquad \forall k$$

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

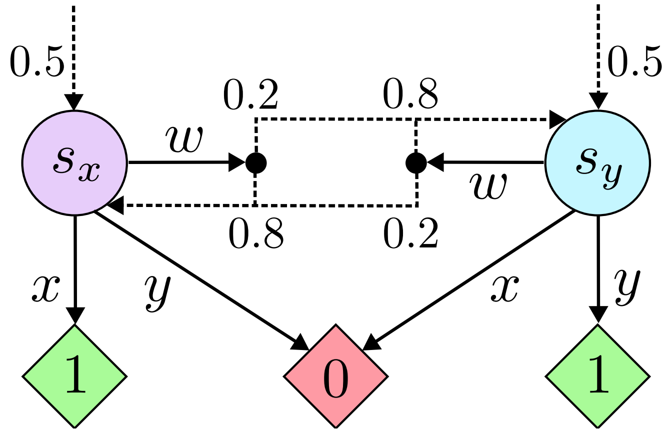
Silver, Veness: *Monte-Carlo Planning in Large POMDPs*, NIPS 2010



- Monte-Carlo algorithm for online planning in large POMDPs
- Combines a Monte-Carlo update of the agent's belief state with a Monte-Carlo tree search from the current belief state.
- Yields new Partially Observable Monte-Carlo Planning (POMCP) algorithm

5. Efficient Approximation

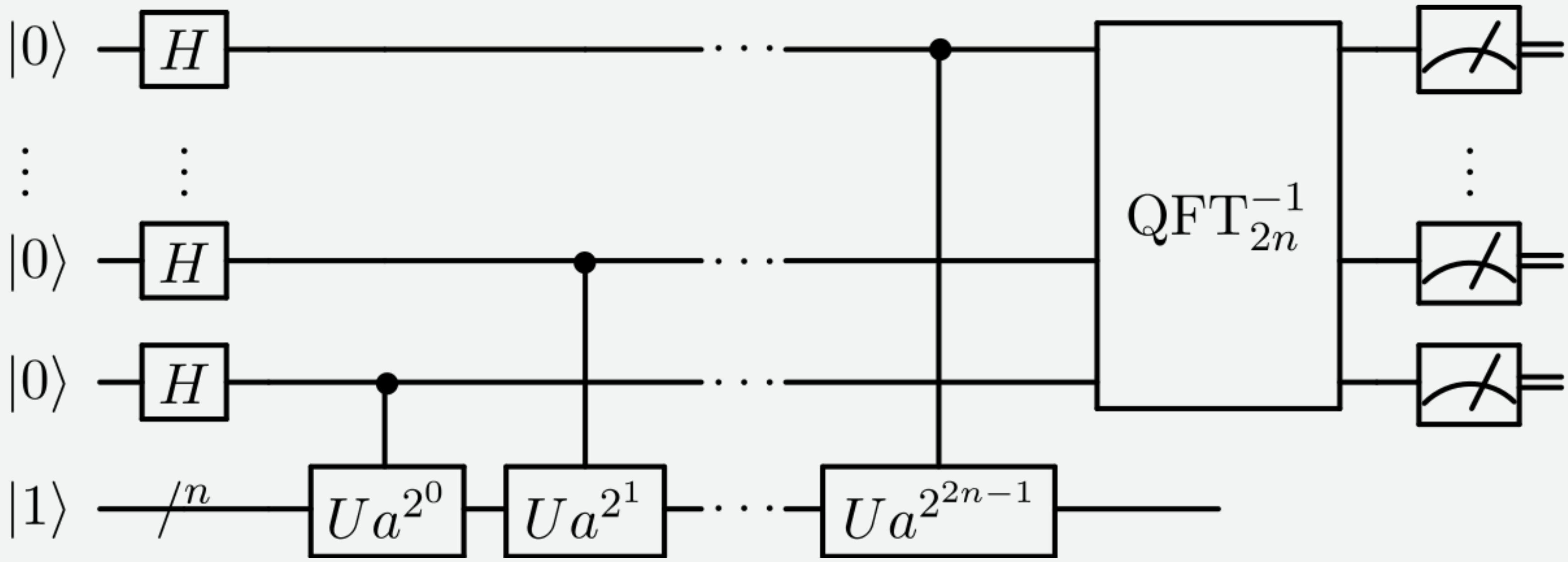
Krale, Koops, Junges, Simão, Jansen: *Tighter Value-Function Approximations for POMDPs*, AAMAS 2025



- Problem: Solving POMDPs typically requires reasoning about exponentially many state beliefs
- State-of-the-art solvers use value bounds to guide reasoning
- Sound and tight upper value bounds often computationally expensive to compute
- Paper introduces new and provably tighter upper value bounds

Outline

A Quantum Program



[Shor‘s Algorithm, Wikipedia]

1. Detecting Bugs using QChecker

- 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
- Bug 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)
```

2. Detecting Bugs using LintQ

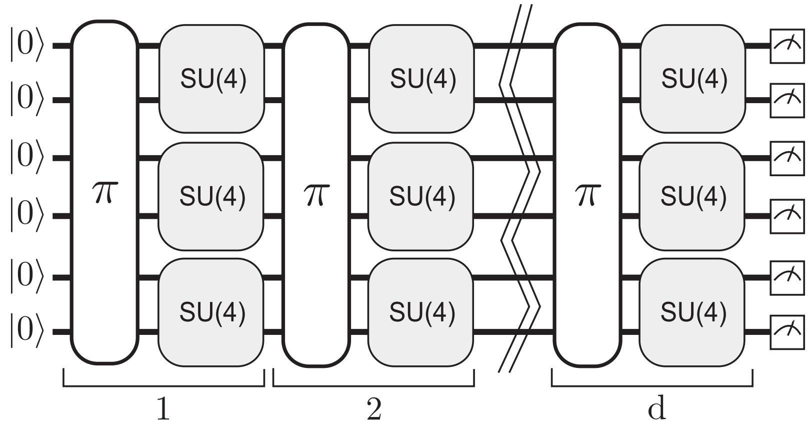
```python
qc = QuantumCircuit(2, 2)
qc.h(1)
qc.cx(1, 0)
qc.measure(0, 0)
qc.measure(1, 1)
qc.z(0) # Problem: Qubit 0 has collapsed
qc.measure(0, 0)
```

```
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, ...

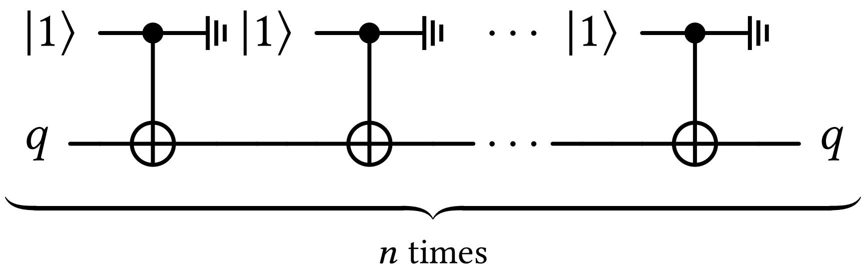
3. The Quantum Volume

- Andrew W. Cross, Lev S. Bishop, Sarah Sheldon, Paul D. Nation, Jay M. Gambetta: *Validating quantum computers using randomized model circuits*, Physical Review A, 2019
- Goal: holistic, single-number metric (quantum volume) that quantifies the largest random circuit of equal width and depth that the computer successfully implements
- Takes qubit coherence times and operational error rates into account
- High-fidelity operations, high connectivity, large calibrated gate sets increase quantum volume



4. Resource Estimation

```
iter :: (Qubit -> Circ Qubit) -> Int
            -> Qubit -> Circ Qubit
iter f 0 q = return q
iter f n q = do
  q <- f q
  iter f (n-1) q
```



- Andrea Colledan, Ugo Dal Lago: *Flexible Type-Based Resource Estimation in Quantum Circuit Description Languages*, POPL 2025
- Type system for Quipper language to derive upper bounds on the size of the circuits compiled from the program
- Can be measured according to various metrics (width, depth, gate count, ...)

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!