UNIVERSITY OF TWENTE.

Importance sampling for dynamic fault trees

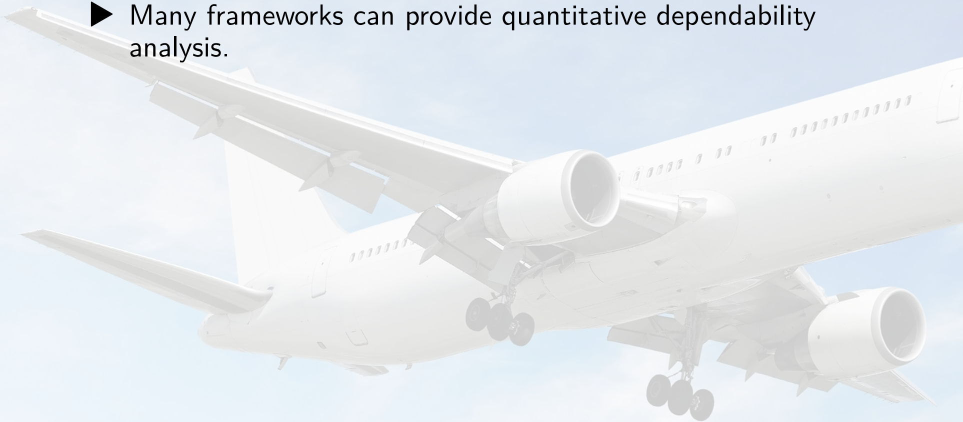
Enno Ruijters, Daniël Reijsbergen,
Pieter-Tjerk de Boer, and Mariëlle
Stoelinga

University of Twente

18 October 2018

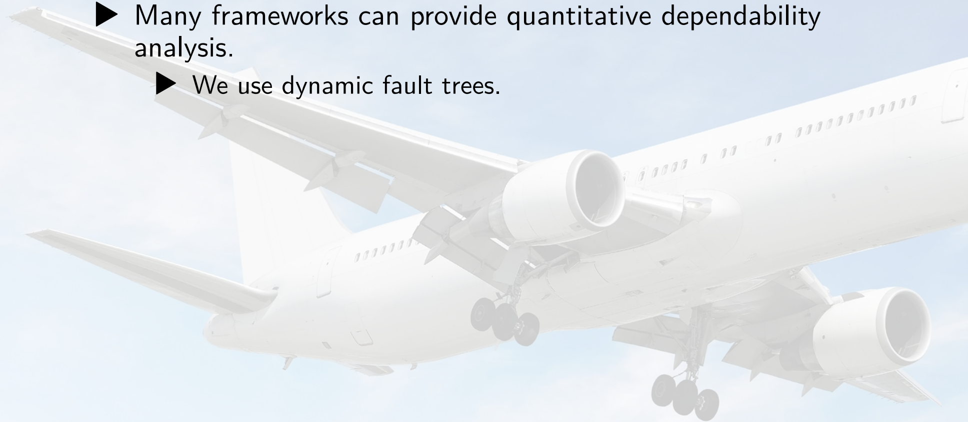
Our contribution in a nutshell

- Many frameworks can provide quantitative dependability analysis.

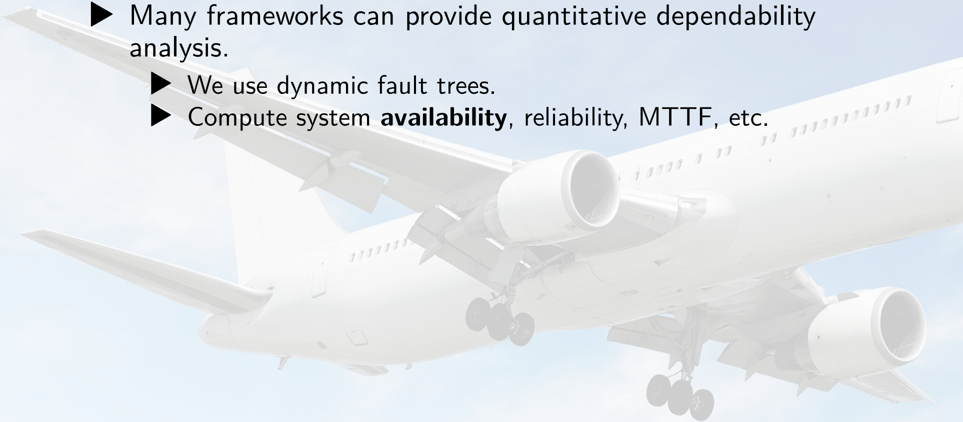


Our contribution in a nutshell

- Many frameworks can provide quantitative dependability analysis.
  - We use dynamic fault trees.



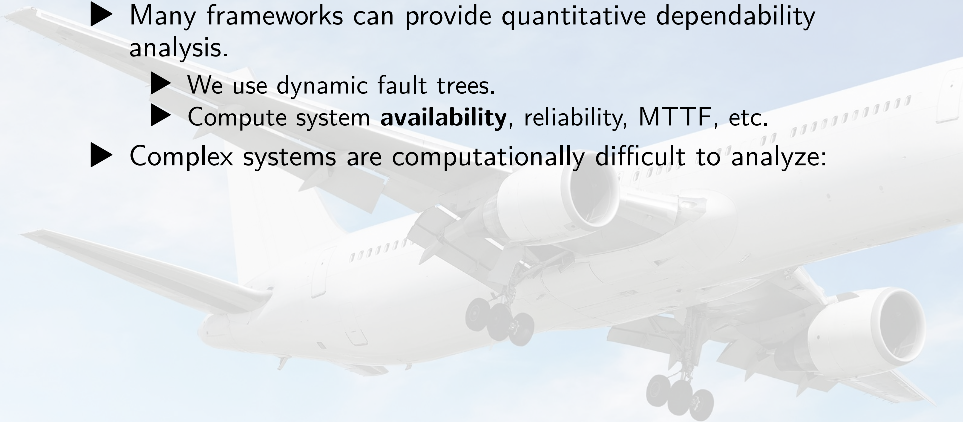
Our contribution in a nutshell



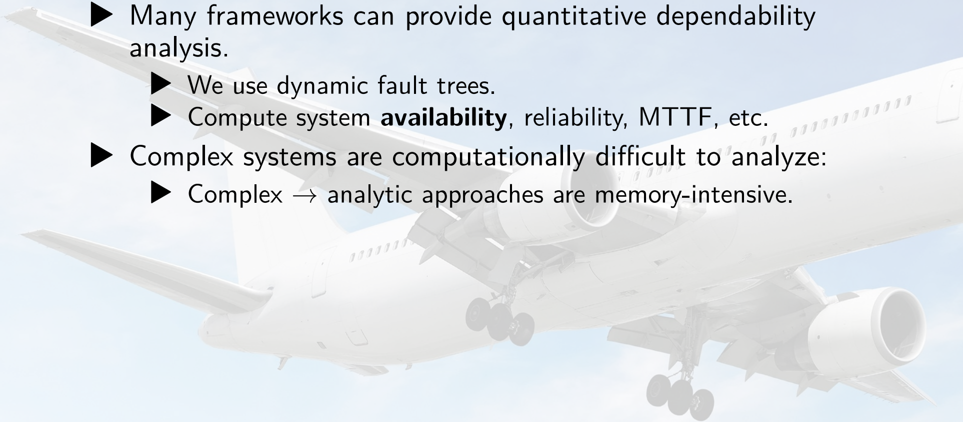
- Many frameworks can provide quantitative dependability analysis.
  - We use dynamic fault trees.
  - Compute system **availability**, reliability, MTTF, etc.

Our contribution in a nutshell

- Many frameworks can provide quantitative dependability analysis.
  - We use dynamic fault trees.
  - Compute system **availability**, reliability, MTTF, etc.
- Complex systems are computationally difficult to analyze:

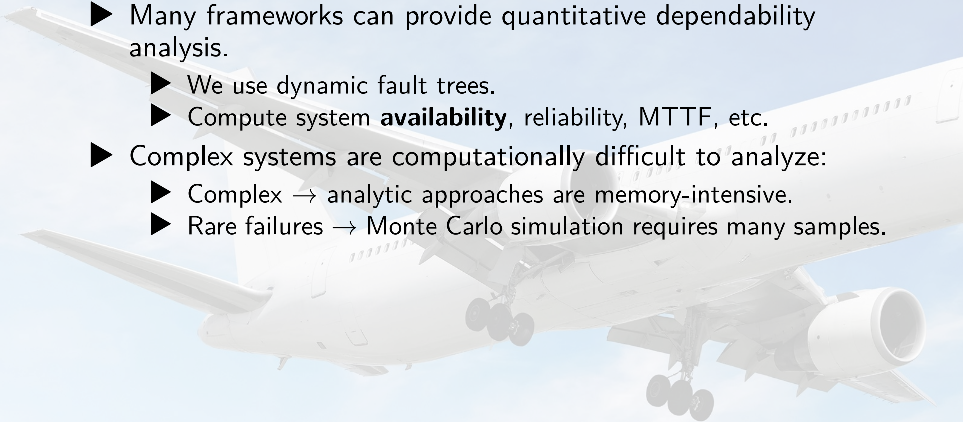


Our contribution in a nutshell



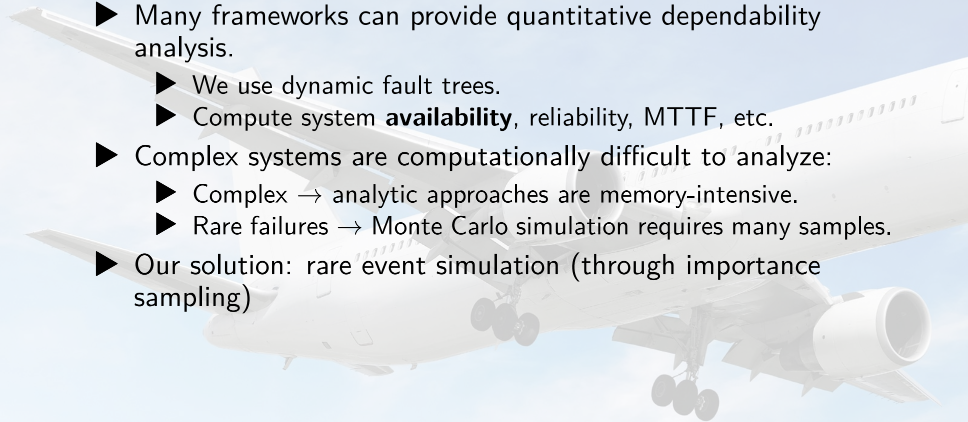
- Many frameworks can provide quantitative dependability analysis.
  - We use dynamic fault trees.
  - Compute system **availability**, reliability, MTTF, etc.
- Complex systems are computationally difficult to analyze:
  - Complex → analytic approaches are memory-intensive.

Our contribution in a nutshell



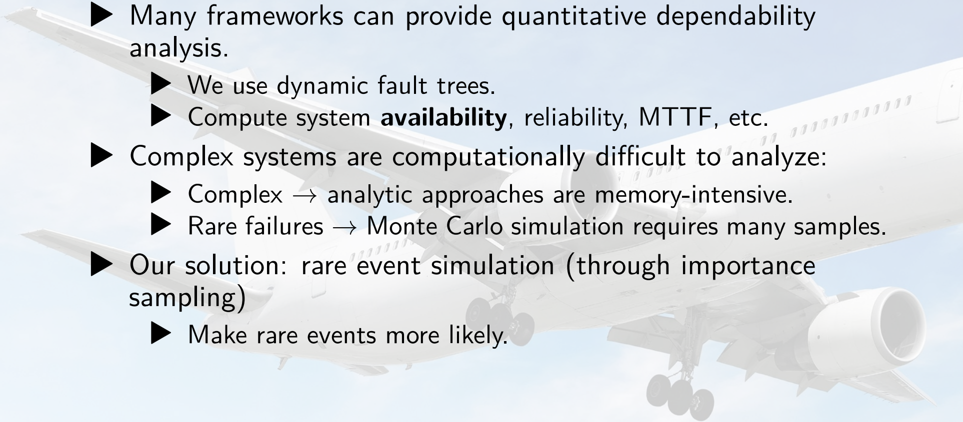
- Many frameworks can provide quantitative dependability analysis.
  - We use dynamic fault trees.
  - Compute system **availability**, reliability, MTTF, etc.
- Complex systems are computationally difficult to analyze:
  - Complex → analytic approaches are memory-intensive.
  - Rare failures → Monte Carlo simulation requires many samples.

Our contribution in a nutshell



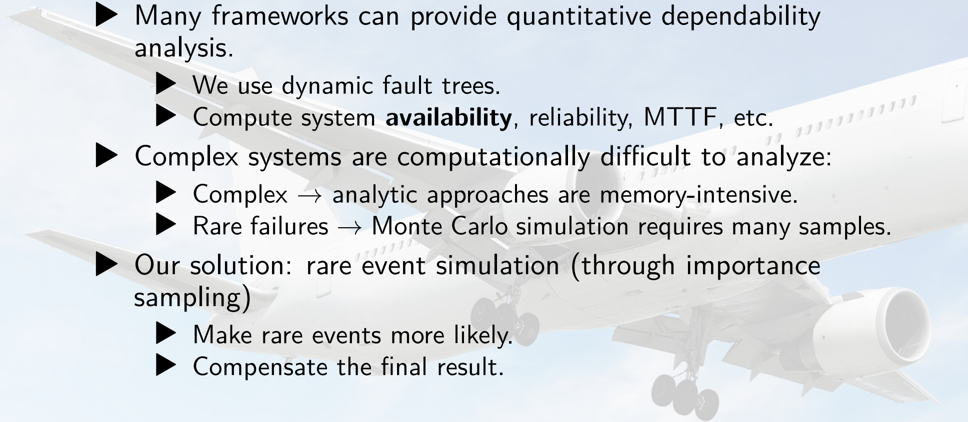
- Many frameworks can provide quantitative dependability analysis.
    - We use dynamic fault trees.
    - Compute system **availability**, reliability, MTTF, etc.
- Complex systems are computationally difficult to analyze:
    - Complex → analytic approaches are memory-intensive.
    - Rare failures → Monte Carlo simulation requires many samples.
- Our solution: rare event simulation (through importance sampling)

Our contribution in a nutshell



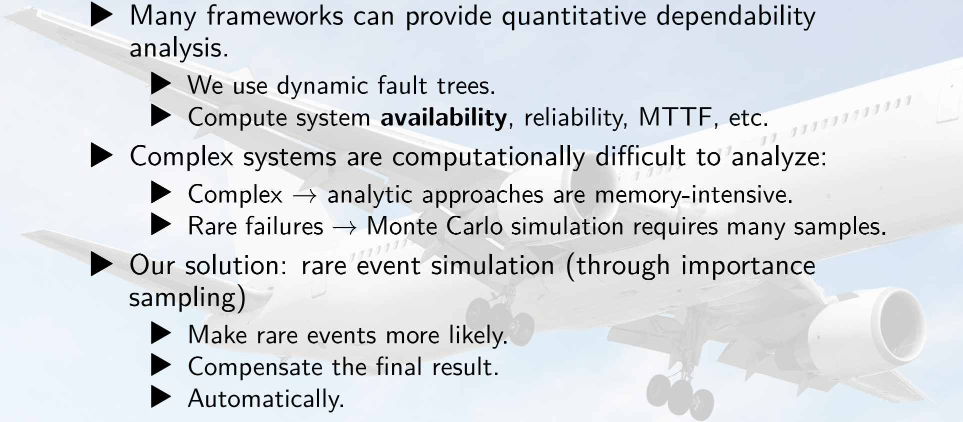
- Many frameworks can provide quantitative dependability analysis.
  - We use dynamic fault trees.
  - Compute system **availability**, reliability, MTTF, etc.
- Complex systems are computationally difficult to analyze:
  - Complex → analytic approaches are memory-intensive.
  - Rare failures → Monte Carlo simulation requires many samples.
- Our solution: rare event simulation (through importance sampling)
  - Make rare events more likely.

Our contribution in a nutshell



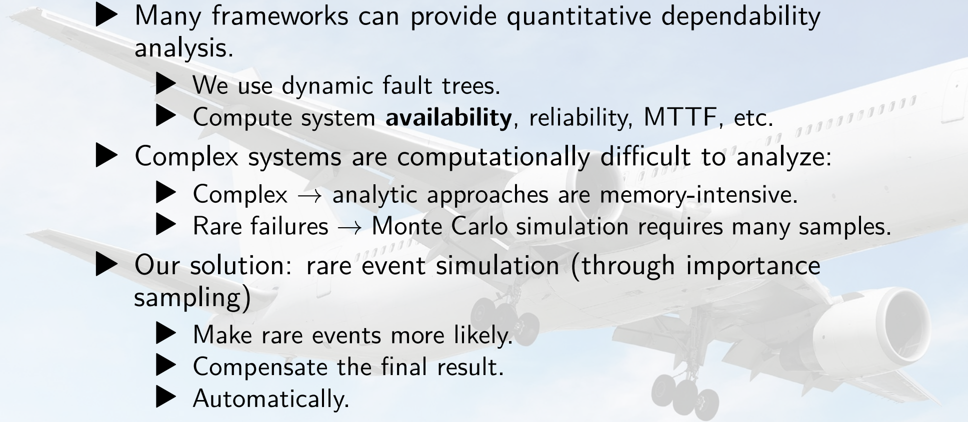
- Many frameworks can provide quantitative dependability analysis.
  - We use dynamic fault trees.
  - Compute system **availability**, reliability, MTTF, etc.
- Complex systems are computationally difficult to analyze:
  - Complex → analytic approaches are memory-intensive.
  - Rare failures → Monte Carlo simulation requires many samples.
- Our solution: rare event simulation (through importance sampling)
  - Make rare events more likely.
  - Compensate the final result.

Our contribution in a nutshell



- Many frameworks can provide quantitative dependability analysis.
  - We use dynamic fault trees.
  - Compute system **availability**, reliability, MTTF, etc.
- Complex systems are computationally difficult to analyze:
  - Complex → analytic approaches are memory-intensive.
  - Rare failures → Monte Carlo simulation requires many samples.
- Our solution: rare event simulation (through importance sampling)
  - Make rare events more likely.
  - Compensate the final result.
  - Automatically.

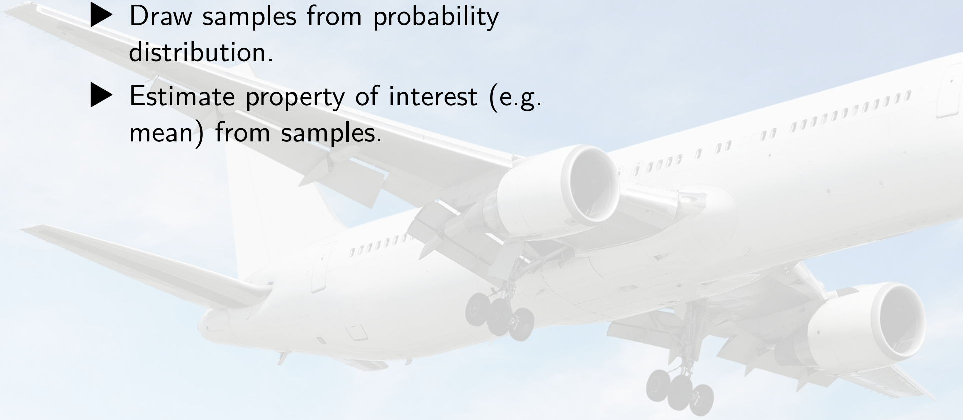
Our contribution in a nutshell



- Many frameworks can provide quantitative dependability analysis.
  - We use dynamic fault trees.
  - Compute system **availability**, reliability, MTTF, etc.
- Complex systems are computationally difficult to analyze:
  - Complex → analytic approaches are memory-intensive.
  - Rare failures → Monte Carlo simulation requires many samples.
- Our solution: rare event simulation (through importance sampling)
  - Make rare events more likely.
  - Compensate the final result.
  - Automatically.
- **Rare event simulation + dynamic fault trees → Faster/more accurate fault tree simulation.**

Monte carlo simulation

- Draw samples from probability distribution.
- Estimate property of interest (e.g. mean) from samples.



Monte carlo simulation

- Draw samples from probability distribution.
- Estimate property of interest (e.g. mean) from samples.
- Example:
  - Spin roulette wheel 1000 times.
  - Observe 36 times green outcome (95% CI boundary).
  - Estimate 3.6% probability of green.
  - (Actual: $\frac{1}{37} = 2.7\%$).

Monte carlo simulation

- Draw samples from probability distribution.
- Estimate property of interest (e.g. mean) from samples.
- Example:
  - Spin roulette wheel 1000 times.
  - Observe 36 times green outcome (95% CI boundary).
  - Estimate 3.6% probability of green.
  - (Actual: $\frac{1}{37} = 2.7\%$).
- Drawback: For improbable events, many samples are needed.

Rare event simulation: Importance sampling

- To reduce required samples: Adjust probabilities and compensate result (Change of Measure).
- Make rare events less rare.

Rare event simulation: Importance sampling

- To reduce required samples: Adjust probabilities and compensate result (Change of Measure).
- Make rare events less rare.
- Example:
    - Spin American roulette wheel 1000 times.
    - Observe 65 times green outcome (95% CI boundary).
    - Estimate 6.5% probability of green **in adjusted system**.
    - (Actual: $\frac{2}{38} = 5.3\%$).

Rare event simulation: Importance sampling

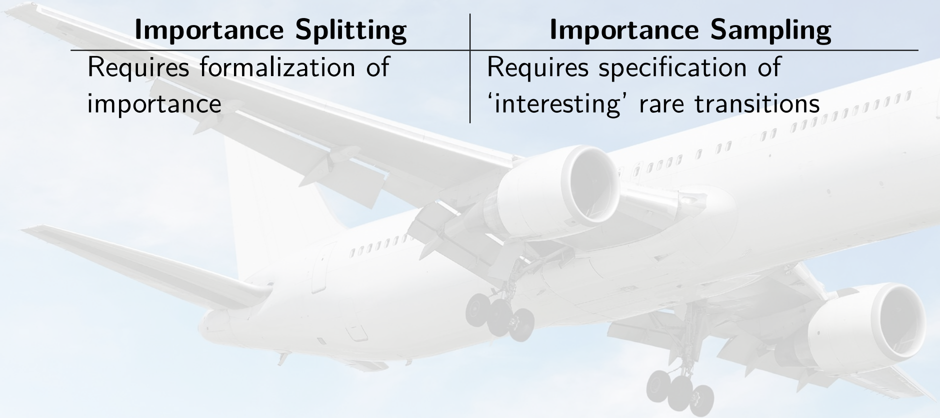
- To reduce required samples: Adjust probabilities and compensate result (Change of Measure).
- Make rare events less rare.
- Example:
  - Spin American roulette wheel 1000 times.
  - Observe 65 times green outcome (95% CI boundary).
  - Estimate 6.5% probability of green **in adjusted system**.
  - (Actual: $\frac{2}{38} = 5.3\%$).
  - Estimate 3.3% probability of green **in original system**.
  - (Actual: $\frac{1}{37} = 2.7\%$).

Rare event simulation: Importance sampling

- To reduce required samples: Adjust probabilities and compensate result (Change of Measure).
- Make rare events less rare.
- Example:
  - Spin American roulette wheel 1000 times.
  - Observe 65 times green outcome (95% CI boundary).
  - Estimate 6.5% probability of green **in adjusted system**.
  - (Actual: $\frac{2}{38} = 5.3\%$).
  - Estimate 3.3% probability of green **in original system**.
  - (Actual: $\frac{1}{37} = 2.7\%$).

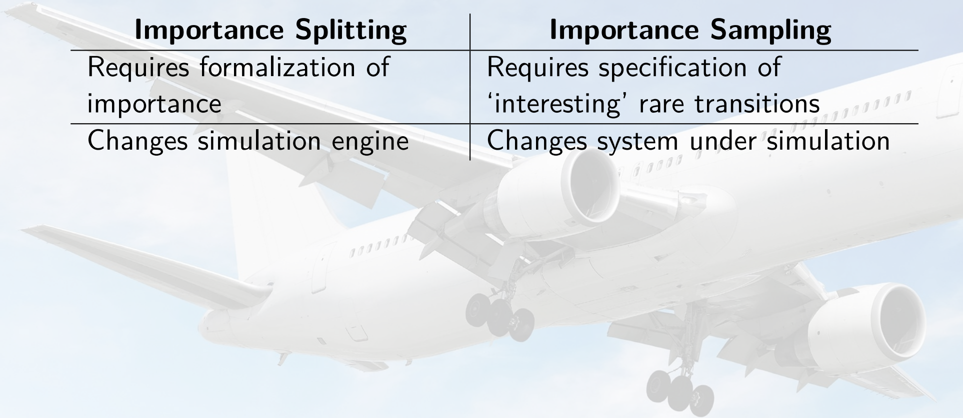
- Yields more accurate results and/or needs fewer samples.

Comparison of RES techniques



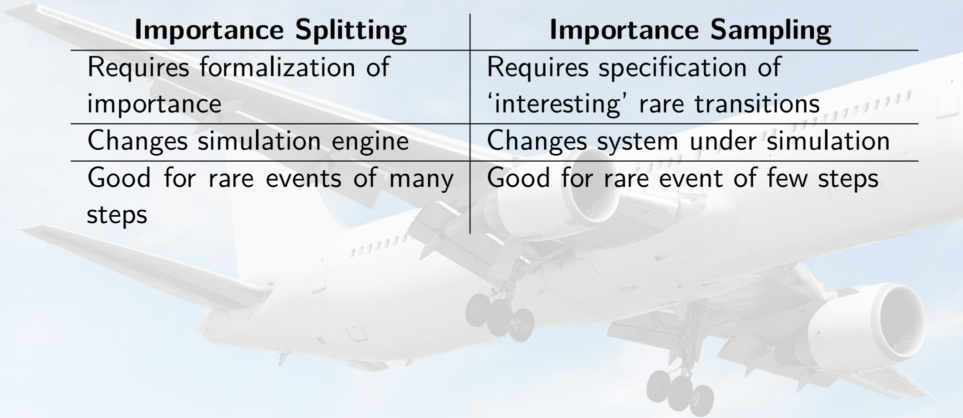
Importance Splitting	**Importance Sampling**
Requires formalization of importance	Requires specification of ‘interesting’ rare transitions

Comparison of RES techniques



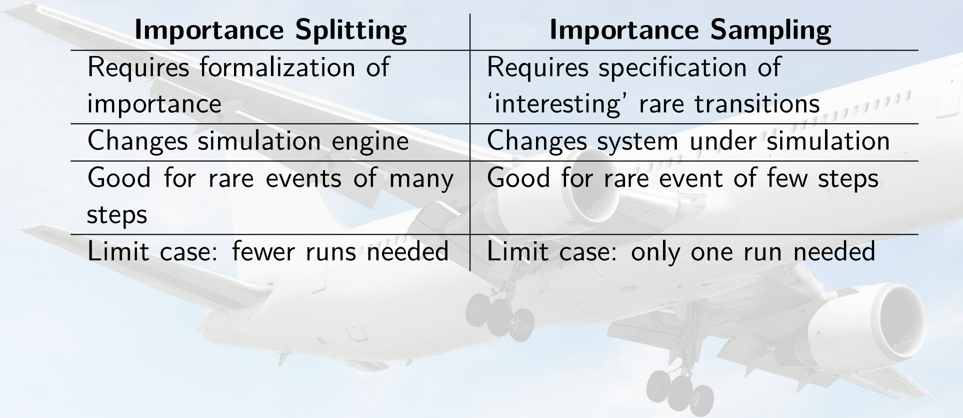
Importance Splitting	**Importance Sampling**
Requires formalization of importance	Requires specification of 'interesting' rare transitions
Changes simulation engine	Changes system under simulation

Comparison of RES techniques



Importance Splitting	**Importance Sampling**
Requires formalization of importance	Requires specification of 'interesting' rare transitions
Changes simulation engine	Changes system under simulation
Good for rare events of many steps	Good for rare event of few steps

Comparison of RES techniques



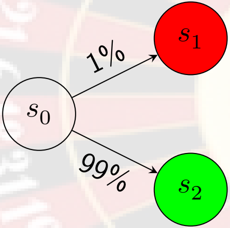
Importance Splitting	**Importance Sampling**
Requires formalization of importance	Requires specification of 'interesting' rare transitions
Changes simulation engine	Changes system under simulation
Good for rare events of many steps	Good for rare event of few steps
Limit case: fewer runs needed	Limit case: only one run needed

Comparison of RES techniques

Importance Splitting	**Importance Sampling**
Requires formalization of importance	Requires specification of 'interesting' rare transitions
Changes simulation engine	Changes system under simulation
Good for rare events of many steps	Good for rare event of few steps
Limit case: fewer runs needed	Limit case: only one run needed

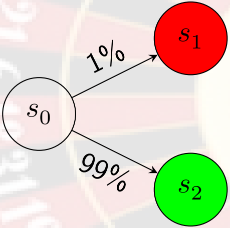
We use importance sampling as our system reaches the rare event after only a few, low-probability transitions. Such models provide few points to split the samples.

Importance sampling of Markov chains: an example



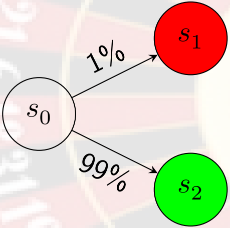
- Probability of red state?

Importance sampling of Markov chains: an example



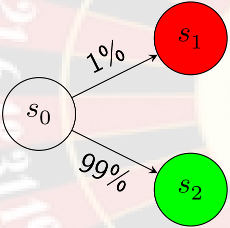
- Probability of red state?
- Estimate 1 (100 runs): 0%

Importance sampling of Markov chains: an example



- Probability of red state?
- Estimate 1 (100 runs): 0%
- Estimate 2 (100 runs): 1%

Importance sampling of Markov chains: an example

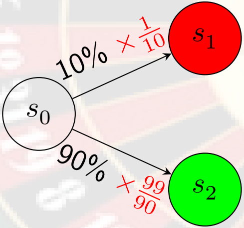


- Probability of red state?
- Estimate 1 (100 runs): 0%
- Estimate 2 (100 runs): 1%
- Estimate 3 (100 runs): 1%
- Estimate 4 (100 runs): 1%
- Estimate 5 (100 runs): 0%
- Estimate 6 (100 runs): 0%
- Estimate 7 (100 runs): 2%

Importance sampling of Markov chains: an example

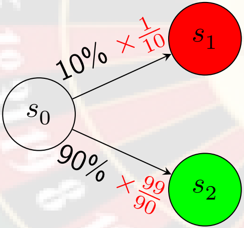
Make bad state 10 times more likely:

- Probability of red state?



Importance sampling of Markov chains: an example

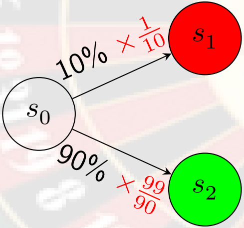
Make bad state 10 times more likely:



- Probability of red state?
- Estimate 1 (100 runs): $10 \cdot \frac{1}{10} = 1\%$

Importance sampling of Markov chains: an example

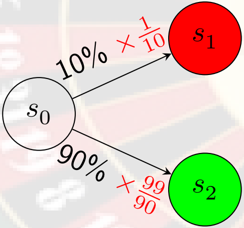
Make bad state 10 times more likely:



- Probability of red state?
- Estimate 1 (100 runs): $10 \cdot \frac{1}{10} = 1\%$
- Estimate 2 (100 runs): $15 \cdot \frac{1}{10} = 1.5\%$

Importance sampling of Markov chains: an example

Make bad state 10 times more likely:

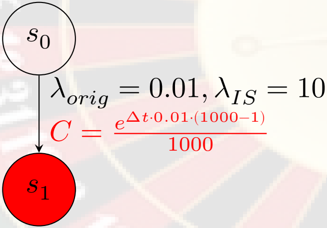


- Probability of red state?
- Estimate 1 (100 runs): $10 \cdot \frac{1}{10} = 1\%$
- Estimate 2 (100 runs): $15 \cdot \frac{1}{10} = 1.5\%$
- Estimate 3 (100 runs): $6 \cdot \frac{1}{10} = 0.6\%$
- Estimate 4 (100 runs): $10 \cdot \frac{1}{10} = 1\%$
- Estimate 5 (100 runs): $13 \cdot \frac{1}{10} = 1.3\%$
- Estimate 6 (100 runs): $8 \cdot \frac{1}{10} = 0.8\%$
- Estimate 7 (100 runs): $14 \cdot \frac{1}{10} = 1.4\%$

Importance sampling of CTMCs: overcompensating

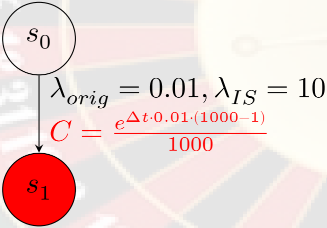
Increasing the rare event too much:

- Probability of reaching red state within 1 time unit?



Importance sampling of CTMCs: overcompensating

Increasing the rare event too much:



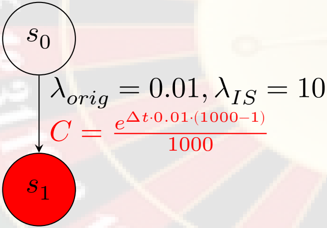
$\lambda_{orig} = 0.01, \lambda_{IS} = 10$

$C = \frac{e^{\Delta t \cdot 0.01 \cdot (1000-1)}}{1000}$

- Probability of reaching red state within 1 time unit?
- Estimate 1 (100 runs): 0.64%
- Estimate 2 (100 runs): 0.45%
- Estimate 3 (100 runs): 0.85%
- Estimate 4 (100 runs): 0.86%

Importance sampling of CTMCs: overcompensating

Increasing the rare event too much:



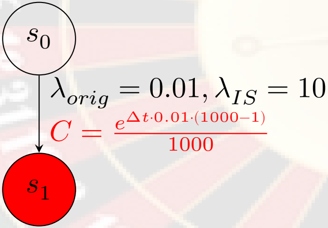
$\lambda_{orig} = 0.01, \lambda_{IS} = 10$

$C = \frac{e^{\Delta t \cdot 0.01 \cdot (1000-1)}}{1000}$

- Probability of reaching red state within 1 time unit?
- Estimate 1 (100 runs): 0.64%
- Estimate 2 (100 runs): 0.45%
- Estimate 3 (100 runs): 0.85%
- Estimate 4 (100 runs): 0.86%
- Estimate 6 (100 runs): 11.26%

Importance sampling of CTMCs: overcompensating

Increasing the rare event too much:



- Probability of reaching red state within 1 time unit?
- Estimate 1 (100 runs): 0.64%
- Estimate 2 (100 runs): 0.45%
- Estimate 3 (100 runs): 0.85%
- Estimate 4 (100 runs): 0.86%
- Estimate 6 (100 runs): 11.26%
- Estimate 7 (100 runs): 0.67%

Path-ZVA algorithm

- Importance sampling algorithm for Markovian models.

Path-ZVA algorithm

- Importance sampling algorithm for Markovian models.
- General concept:
    - Identify shortest paths to goal state.
    - Prioritize transitions following this (small part of) the state space.
    - Return to standard Monte Carlo simulation off the dominant paths.

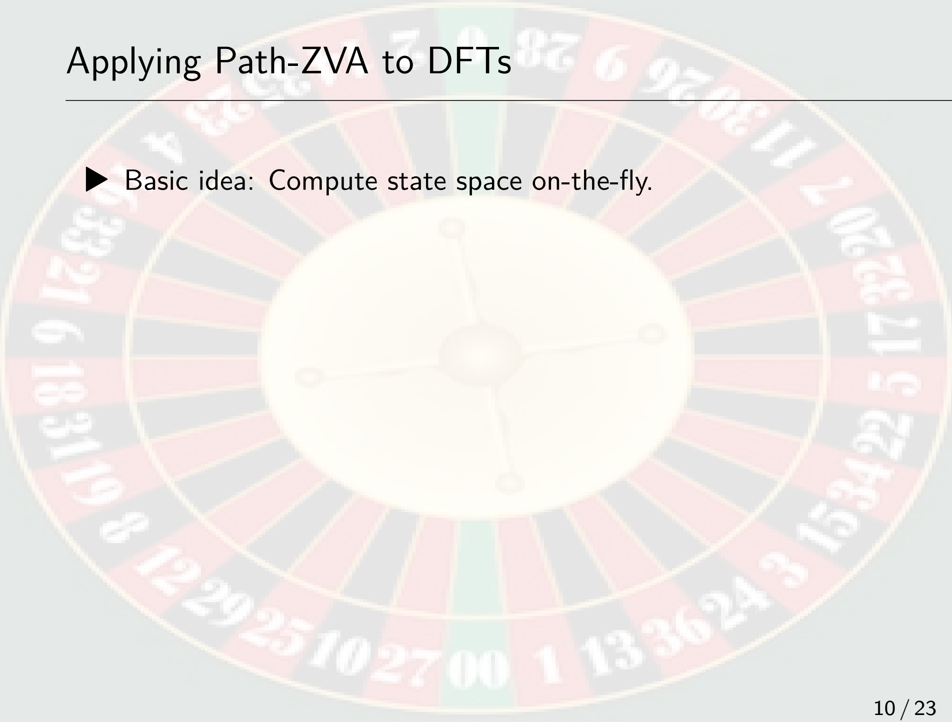
Path-ZVA algorithm

- Importance sampling algorithm for Markovian models.
- General concept:
  - Identify shortest paths to goal state.
  - Prioritize transitions following this (small part of) the state space.
  - Return to standard Monte Carlo simulation off the dominant paths.
- Estimates:
  - Probability of reaching goal states before reaching to taboo states.
  - Fraction of time spend in goal states for cyclic models.
  - WIP: Time-bounded probability of reaching goal states.

Path-ZVA algorithm

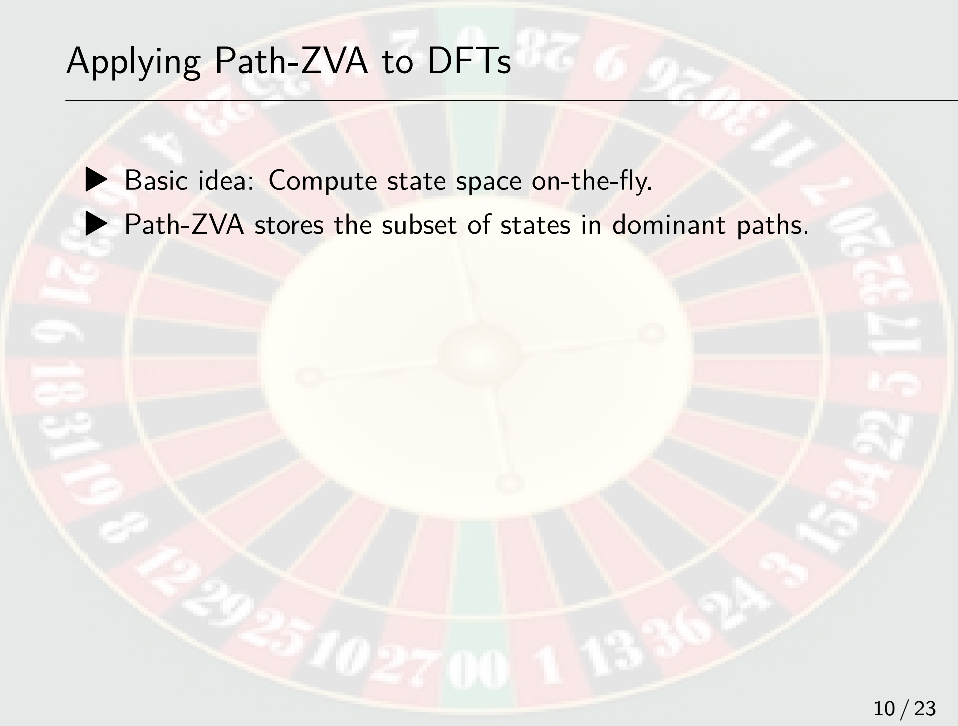
- Importance sampling algorithm for Markovian models.
- General concept:
  - Identify shortest paths to goal state.
  - Prioritize transitions following this (small part of) the state space.
  - Return to standard Monte Carlo simulation off the dominant paths.
- Estimates:
  - Probability of reaching goal states before reaching to taboo states.
  - Fraction of time spend in goal states for cyclic models.
  - WIP: Time-bounded probability of reaching goal states.
- Transition rates parameterized as $r \cdot \epsilon^n$ with $0 < \epsilon << 1$ to indicate ‘rareness’.

Applying Path-ZVA to DFTs



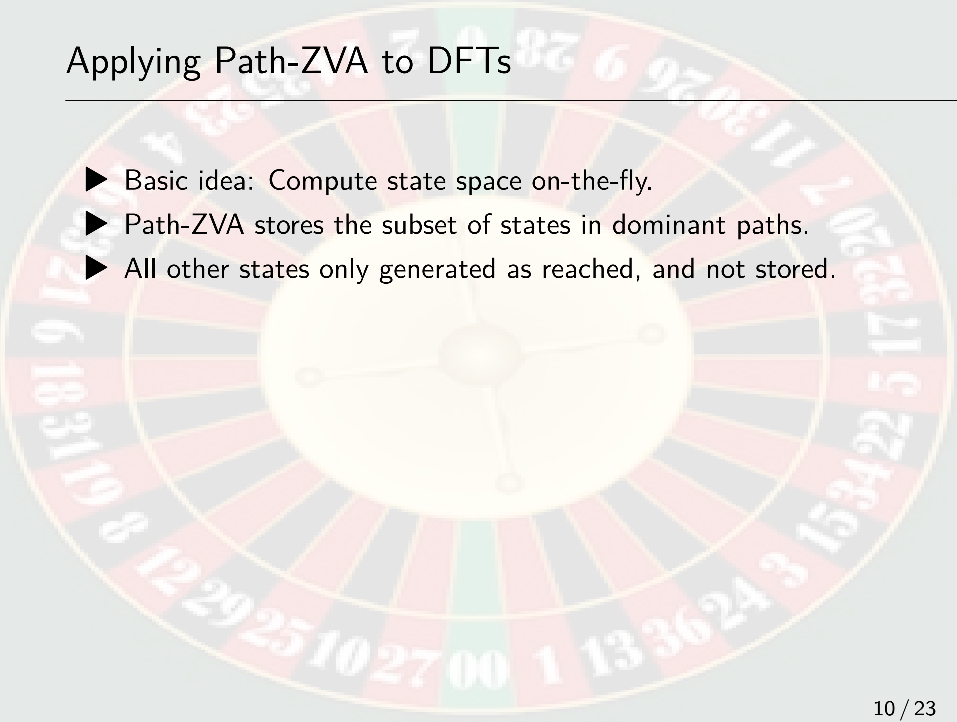
- Basic idea: Compute state space on-the-fly.

Applying Path-ZVA to DFTs



- Basic idea: Compute state space on-the-fly.
- Path-ZVA stores the subset of states in dominant paths.

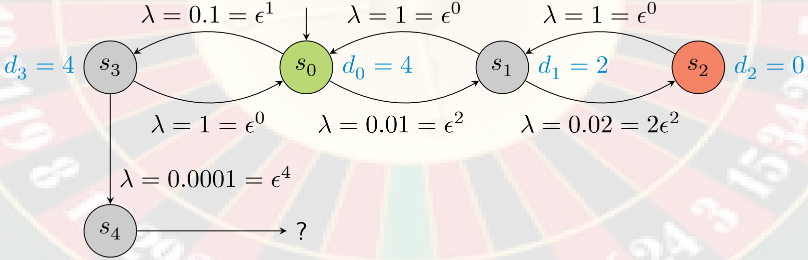
Applying Path-ZVA to DFTs



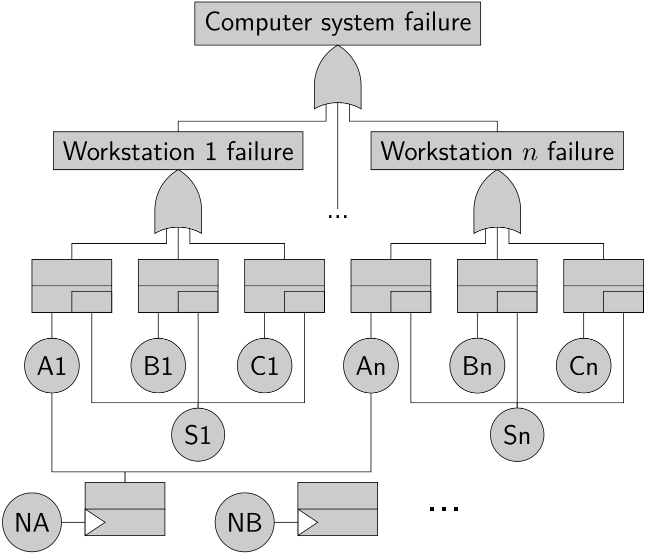
- Basic idea: Compute state space on-the-fly.
- Path-ZVA stores the subset of states in dominant paths.
- All other states only generated as reached, and not stored.

Applying Path-ZVA to DFTs

- Basic idea: Compute state space on-the-fly.
- Path-ZVA stores the subset of states in dominant paths.
- All other states only generated as reached, and not stored.



DFT example


Computer system failure
Workstation 1 failure
Workstation n failure
...
A1
B1
C1
An
Bn
Cn
S1
Sn
NA
NB
...

From I/O-IMCs to Markov Chains

- Path-ZVA requires (continuous-time) Markov chains.
- DFTCalc produces I/O-IMCs.

From I/O-IMCs to Markov Chains

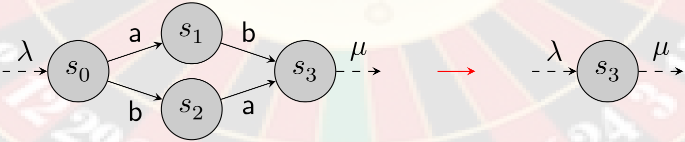
- Path-ZVA requires (continuous-time) Markov chains.
- DFTCalc produces I/O-IMCs.
- How to resolve nondeterminism?

From I/O-IMCs to Markov Chains

- Path-ZVA requires (continuous-time) Markov chains.
- DFTCalc produces I/O-IMCs.
- How to resolve nondeterminism?
- Generally, the nondeterminism is spurious:
    - Regardless of choices, you end up in the same Markovian state.

From I/O-IMCs to Markov Chains

- Path-ZVA requires (continuous-time) Markov chains.
- DFTCalc produces I/O-IMCs.
- How to resolve nondeterminism?
- Generally, the nondeterminism is spurious:
  - Regardless of choices, you end up in the same Markovian state.
- We verify that it is spurious, then collapse the interactive transitions before the next state:

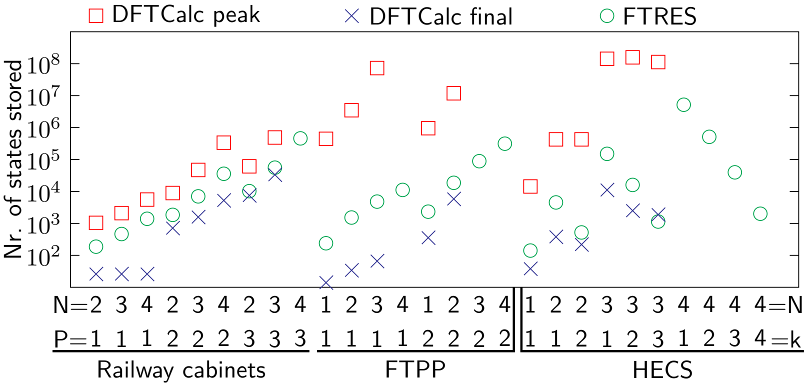


Results: Unavailability

Exact result for DFTCalc, 95% confidence for others:

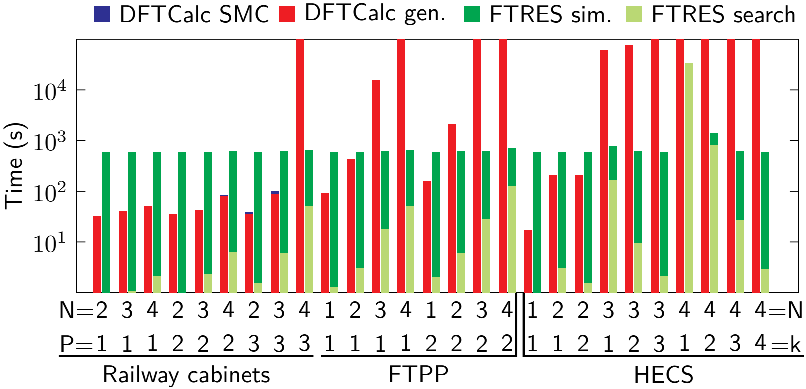
			Unavailability		
	N	**P**	**DFTCalc**	**FTRES**	**MC**
FTPP	1	1	$2.18303 \cdot 10^{-10}$	$[2.182; 2.184] \cdot 10^{-10}$	—
	4	1	—	$[2.226; 2.232] \cdot 10^{-10}$	—
	1	2	$1.76174 \cdot 10^{-20}$	$[1.761; 1.762] \cdot 10^{-20}$	—
	4	2	—	$[1.760; 1.763] \cdot 10^{-20}$	—
	N	**k**	**DFTCalc**	**FTRES**	**MC**
HECS	1	1	$4.12485 \cdot 10^{-5}$	$[4.124; 4.126] \cdot 10^{-5}$	$[4.079; 4.156] \cdot 10^{-5}$
	2	1	$3.02469 \cdot 10^{-9}$	$[3.022; 3.026] \cdot 10^{-9}$	$[0; 9.040] \cdot 10^{-9}$
	2	2	$8.24940 \cdot 10^{-5}$	$[8.247; 8.251] \cdot 10^{-5}$	$[8.218; 8.338] \cdot 10^{-5}$
	4	1	—	$[3.902; 4.364] \cdot 10^{-17}$	—
	4	2	—	$[1.239; 1.252] \cdot 10^{-12}$	—
	4	3	—	$[1.813; 1.818] \cdot 10^{-8}$	$[0; 8.352] \cdot 10^{-9}$
	4	4	—	$[1.648; 1.651] \cdot 10^{-4}$	$[1.621; 1.657] \cdot 10^{-4}$

Overall results: State space



- FTRES always below DFTCalc maximal state space size.
- FTRES computes results where DFTCalc does not.

Overall results: Speed



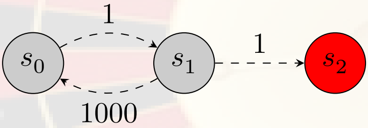
- FTRes and MC spend a constant 10 mins. simulating.
- Simulation time mostly dominates state-space exploration.
- Several DFTCalc experiments ran longer than 24 hours.

WIP: Extending to time-bounded reachability

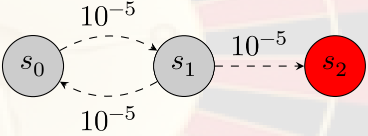
- Standard Path-ZVA leaves total exit rates unchanged.

WIP: Extending to time-bounded reachability

- Standard Path-ZVA leaves total exit rates unchanged.
- Two causes of rarity with time bounds:



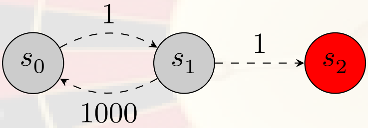
Rarity due to
relative probability



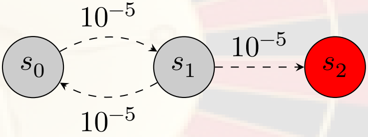
Rarity due to
exit rates

WIP: Extending to time-bounded reachability

- Standard Path-ZVA leaves total exit rates unchanged.
- Two causes of rarity with time bounds:



Rarity due to
relative probability



Rarity due to
exit rates

- Path-ZVA helps the first case, not the second.

WIP: Extending to time-bounded reachability

Various possible compensations for rate-caused rarity:

- *Fixed acceleration*: Multiply all exit rates by the same factor.

WIP: Extending to time-bounded reachability

Various possible compensations for rate-caused rarity:

- *Fixed acceleration*: Multiply all exit rates by the same factor.
  - Most intuitive approach.
  - Choice of multiplier is not obvious.
  - Bad multiplier can give misleading results.

WIP: Extending to time-bounded reachability

Various possible compensations for rate-caused rarity:

- *Fixed acceleration*: Multiply all exit rates by the same factor.
    - Most intuitive approach.
    - Choice of multiplier is not obvious.
    - Bad multiplier can give misleading results.
- *Conditioning* or *discrete time conversion*: sample paths ignoring time, then calculate goal probability of each path.

WIP: Extending to time-bounded reachability

Various possible compensations for rate-caused rarity:

- *Fixed acceleration*: Multiply all exit rates by the same factor.
  - Most intuitive approach.
  - Choice of multiplier is not obvious.
  - Bad multiplier can give misleading results.
- *Conditioning* or *discrete time conversion*: sample paths ignoring time, then calculate goal probability of each path.
  - Optimal variance for a given path-sampling scheme.
  - Not particularly fast.

WIP: Extending to time-bounded reachability

Various possible compensations for rate-caused rarity:

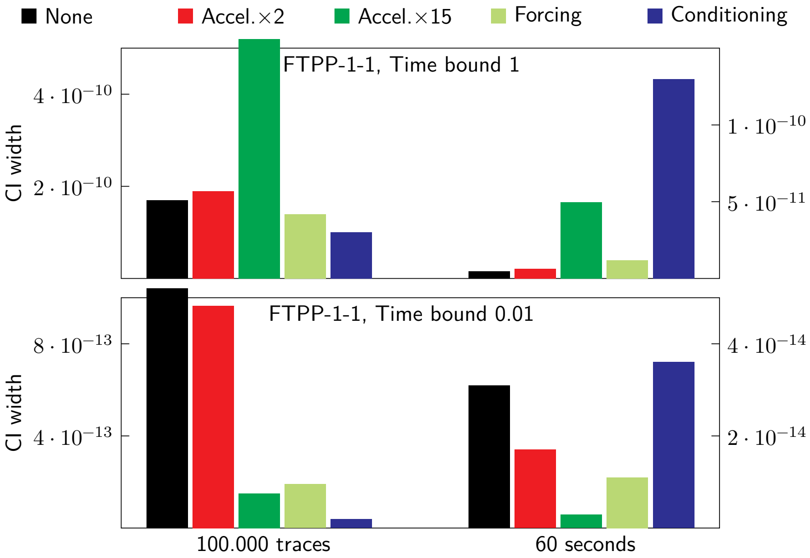
- *Fixed acceleration*: Multiply all exit rates by the same factor.
    - Most intuitive approach.
    - Choice of multiplier is not obvious.
    - Bad multiplier can give misleading results.
- *Conditioning* or *discrete time conversion*: sample paths ignoring time, then calculate goal probability of each path.
    - Optimal variance for a given path-sampling scheme.
    - Not particularly fast.
- *Forcing*: Sample transition times conditional on time bound.

WIP: Extending to time-bounded reachability

Various possible compensations for rate-caused rarity:

- *Fixed acceleration*: Multiply all exit rates by the same factor.
    - Most intuitive approach.
    - Choice of multiplier is not obvious.
    - Bad multiplier can give misleading results.
- *Conditioning* or *discrete time conversion*: sample paths ignoring time, then calculate goal probability of each path.
    - Optimal variance for a given path-sampling scheme.
    - Not particularly fast.
- *Forcing*: Sample transition times conditional on time bound.
    - Pretty good performance in general.
    - Guaranteed better than no forcing (sort of).
    - Moderate loss of performance.

Preliminary results on time-bounded reachability



Preliminary results on time-bounded reachability

Results for FTPP-1-1, other models behave similarly:

Time bound	Forcing /conditioning	CI width 100.000 sims	Simulation time	CI width 60s sim.
1	None	$1.7 \cdot 10^{-10}$	0.16	$4.8 \cdot 10^{-12}$
1	Accel. × 2	$1.9 \cdot 10^{-10}$	0.15	$6.4 \cdot 10^{-12}$
1	Accel. × 15	$1.5 \cdot 10^{-9}$	0.15	$5.0 \cdot 10^{-11}$
1	Forcing	$1.4 \cdot 10^{-10}$	0.60	$1.2 \cdot 10^{-11}$
1	Conditioning	$1.0 \cdot 10^{-10}$	70	$1.3 \cdot 10^{-10}$
0.01	None	$2.1 \cdot 10^{-12}$	0.05	$3.1 \cdot 10^{-14}$
0.01	Accel. × 2	$9.8 \cdot 10^{-13}$	0.08	$1.7 \cdot 10^{-14}$
0.01	Accel. × 15	$1.5 \cdot 10^{-13}$	0.07	$2.9 \cdot 10^{-15}$
0.01	Forcing	$1.9 \cdot 10^{-13}$	0.36	$1.1 \cdot 10^{-14}$
0.01	Conditioning	$3.9 \cdot 10^{-14}$	53	$3.6 \cdot 10^{-14}$

WIP: Extending to time-bounded reachability

Preliminary results:

- Conditioning is too slow to be practical.
  - Usually better to spend the calculation time on more traces.

WIP: Extending to time-bounded reachability

Preliminary results:

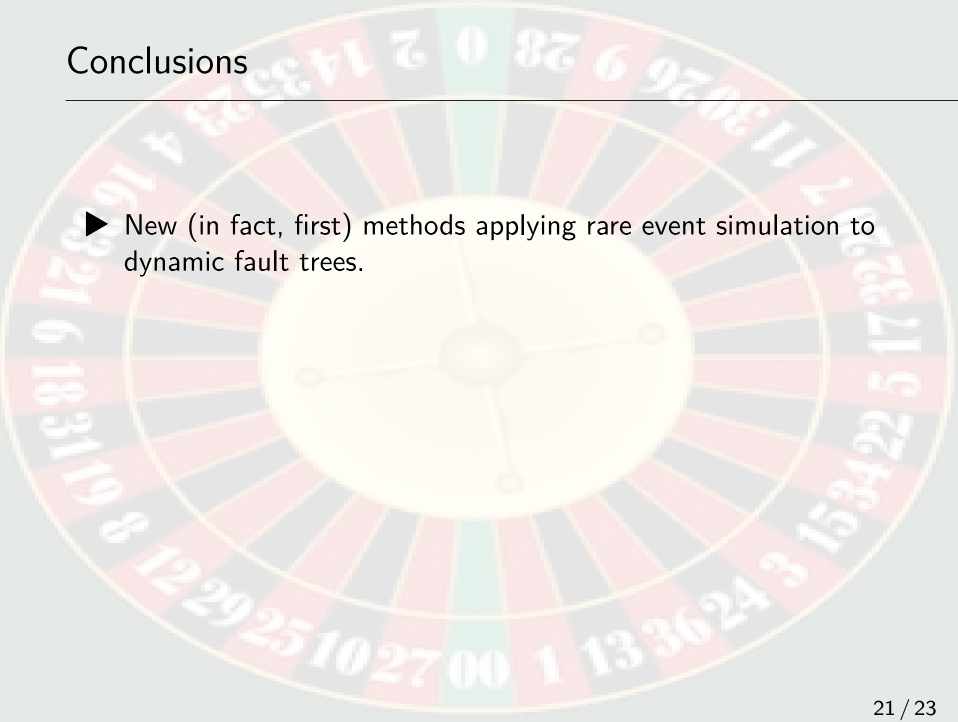
- Conditioning is too slow to be practical.
  - Usually better to spend the calculation time on more traces.
- Fixed acceleration good but fiddly.
  - Good acceleration factors hard to determine in advance.
  - Bad choices can result in biased-looking estimators.

WIP: Extending to time-bounded reachability

Preliminary results:

- Conditioning is too slow to be practical.
  - Usually better to spend the calculation time on more traces.
- Fixed acceleration good but fiddly.
  - Good acceleration factors hard to determine in advance.
  - Bad choices can result in biased-looking estimators.
- Forcing generally works well.
  - Mostly independent of model rates and property.

Conclusions



- New (in fact, first) methods applying rare event simulation to dynamic fault trees.

Conclusions

- New (in fact, first) methods applying rare event simulation to dynamic fault trees.
- On-the-fly composition avoids state-space explosion.

Conclusions

- New (in fact, first) methods applying rare event simulation to dynamic fault trees.
- On-the-fly composition avoids state-space explosion.
- Importance sampling provides tight confidence intervals even for very rare events.

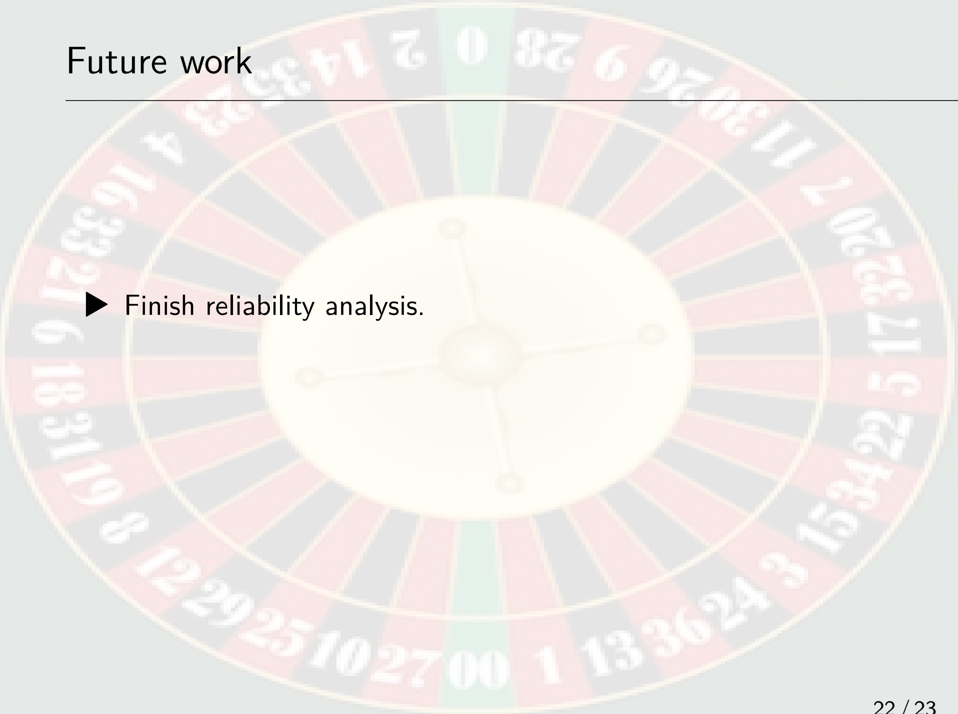
Conclusions

- New (in fact, first) methods applying rare event simulation to dynamic fault trees.
- On-the-fly composition avoids state-space explosion.
- Importance sampling provides tight confidence intervals even for very rare events.
- We handle larger models than DFTCalc, and models of more reliable systems than Monte Carlo simulation.

Conclusions

- New (in fact, first) methods applying rare event simulation to dynamic fault trees.
- On-the-fly composition avoids state-space explosion.
- Importance sampling provides tight confidence intervals even for very rare events.
- We handle larger models than DFTCalc, and models of more reliable systems than Monte Carlo simulation.
- Analysis of availability and reliability, with reliability still being improved.

Future work



- Finish reliability analysis.

Future work

- Finish reliability analysis.
- Support for non-Markovian timing of e.g. maintenance actions.

Future work

- Finish reliability analysis.
- Support for non-Markovian timing of e.g. maintenance actions.
- Limited nondeterminism to cover full set of DFTs

Thank you for your attention.

Questions?