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.

▶ 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.
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 solution: rare event simulation (through importance sampling)
    - Make rare events more likely.
    - Compensate the final result.
- Rare event simulation + dynamic fault trees → Faster/more accurate fault tree simulation.
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 $\rightarrow$ analytic approaches are memory-intensive.
  - Rare failures $\rightarrow$ 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 $\rightarrow$ analytic approaches are memory-intensive.
  - Rare failures $\rightarrow$ 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.

Example:
- Spin roulette wheel 1000 times.
- Observe 36 times green outcome (95% CI boundary).
- Estimate 3.6% probability of green.
  (Actual: 137 = 2.7%).

Drawback: For improbable events, many samples are needed.
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.

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: 238 = 5.3%).
- Estimate 3.3% probability of green in original system.
- (Actual: 137 = 2.7%).

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

<table>
<thead>
<tr>
<th>Importance Splitting</th>
<th>Importance Sampling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requires formalization of importance</td>
<td>Requires specification of ‘interesting’ rare transitions</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.
Comparison of RES techniques

<table>
<thead>
<tr>
<th>Importance Splitting</th>
<th>Importance Sampling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requires formalization of importance</td>
<td>Requires specification of ‘interesting’ rare transitions</td>
</tr>
<tr>
<td>Changes simulation engine</td>
<td>Changes system under simulation</td>
</tr>
</tbody>
</table>

Good for rare events of many steps: requires formalization of importance.

Good for rare event of few steps: requires specification of ‘interesting’ rare transitions.

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.
## Comparison of RES techniques

<table>
<thead>
<tr>
<th>Importance Splitting</th>
<th>Importance Sampling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requires formalization of importance</td>
<td>Requires specification of ‘interesting’ rare transitions</td>
</tr>
<tr>
<td>Changes simulation engine</td>
<td>Changes system under simulation</td>
</tr>
<tr>
<td>Good for rare events of many steps</td>
<td>Good for rare event of few steps</td>
</tr>
</tbody>
</table>

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.
Comparison of RES techniques

<table>
<thead>
<tr>
<th>Importance Splitting</th>
<th>Importance Sampling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requires formalization of importance</td>
<td>Requires specification of ‘interesting’ rare transitions</td>
</tr>
<tr>
<td>Changes simulation engine</td>
<td>Changes system under simulation</td>
</tr>
<tr>
<td>Good for rare events of many steps</td>
<td>Good for rare event of few steps</td>
</tr>
<tr>
<td>Limit case: fewer runs needed</td>
<td>Limit case: only one run needed</td>
</tr>
</tbody>
</table>

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.
Comparison of RES techniques

<table>
<thead>
<tr>
<th>Importance Splitting</th>
<th>Importance Sampling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requires formalization of importance</td>
<td>Requires specification of ‘interesting’ rare transitions</td>
</tr>
<tr>
<td>Changes simulation engine</td>
<td>Changes system under simulation</td>
</tr>
<tr>
<td>Good for rare events of many steps</td>
<td>Good for rare event of few steps</td>
</tr>
<tr>
<td>Limit case: fewer runs needed</td>
<td>Limit case: only one run needed</td>
</tr>
</tbody>
</table>

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?

- $s_0$: 99%
- $s_1$: 1%
- $s_2$: 1%

Estimates (100 runs):
- Estimate 1: 0%
- Estimate 2: 1%
- Estimate 3: 1%
- Estimate 4: 1%
- Estimate 5: 0%
- Estimate 6: 0%
- Estimate 7: 2%
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

- 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

- 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?

$\text{Estimate 1 (100 runs): } 10 \times \frac{1}{10} = 1%$

$\text{Estimate 2 (100 runs): } 15 \times \frac{1}{10} = 1.5%$

$\text{Estimate 3 (100 runs): } 6 \times \frac{1}{10} = 0.6%$

$\text{Estimate 4 (100 runs): } 10 \times \frac{1}{10} = 1%$

$\text{Estimate 5 (100 runs): } 13 \times \frac{1}{10} = 1.3%$

$\text{Estimate 6 (100 runs): } 8 \times \frac{1}{10} = 0.8%$

$\text{Estimate 7 (100 runs): } 14 \times \frac{1}{10} = 1.4%$
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:

\( s_0 \)
\[ \lambda_{\text{orig}} = 0.01, \lambda_{IS} = 10 \]
\[ C = \frac{e^{\Delta t \cdot 0.01 \cdot (1000-1)}}{1000} \]

\( s_1 \)

▶ Probability of reaching red state within 1 time unit?
Importance sampling of CTMCs: overcompensating

Increasing the rare event too much:

\[ \lambda_{\text{orig}} = 0.01, \lambda_{\text{IS}} = 10 \]

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

- 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%
- Estimate 7 (100 runs): 0.67%
Importance sampling of CTMCs: overcompensating

Increasing the rare event too much:

\[
\lambda_{orig} = 0.01, \quad \lambda_{IS} = 10
\]

\[
C = \frac{e^{\lambda_{orig}(1000-1)}}{1000}
\]

- 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:

\[ C = e^{Δt \cdot 0.01 \cdot (1000 - 1)} \frac{1}{1000} \]

\( \lambda_{orig} = 0.01, \lambda_{IS} = 10 \)

- 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 \( r \cdot \varepsilon^n \) with \( 0 < \varepsilon \ll 1 \) 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.

\[ \text{Path-ZVA requires (continuous-time) Markov chains.} \]
\[ \text{DFTCalc produces I/O-IMCs.} \]
From I/O-IMCs to Markov Chains

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

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:
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:

![Diagram](attachment:image.png)
## Results: Unavailability

### Exact result for DFTCalc, 95% confidence for others:

<table>
<thead>
<tr>
<th>N</th>
<th>P</th>
<th>DFTCalc</th>
<th>FTRES</th>
<th>MC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>Unavailability</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>FTTP</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$2.18303 \cdot 10^{-10}$</td>
<td>[2.182; 2.184] $\cdot 10^{-10}$</td>
<td>—</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>—</td>
<td>[2.226; 2.232] $\cdot 10^{-10}$</td>
<td>—</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>$1.76174 \cdot 10^{-20}$</td>
<td>[1.761; 1.762] $\cdot 10^{-20}$</td>
<td>—</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>—</td>
<td>[1.760; 1.763] $\cdot 10^{-20}$</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>HECS</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$4.12485 \cdot 10^{-5}$</td>
<td>[4.124; 4.126] $\cdot 10^{-5}$</td>
<td>[4.079; 4.156] $\cdot 10^{-5}$</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>$3.02469 \cdot 10^{-9}$</td>
<td>[3.022; 3.026] $\cdot 10^{-9}$</td>
<td>[0; 9.040] $\cdot 10^{-9}$</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>$8.24940 \cdot 10^{-5}$</td>
<td>[8.247; 8.251] $\cdot 10^{-5}$</td>
<td>[8.218; 8.338] $\cdot 10^{-5}$</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>—</td>
<td>[3.902; 4.364] $\cdot 10^{-17}$</td>
<td>—</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>—</td>
<td>[1.239; 1.252] $\cdot 10^{-12}$</td>
<td>—</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>—</td>
<td>[1.813; 1.818] $\cdot 10^{-8}$</td>
<td>[0; 8.352] $\cdot 10^{-9}$</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>—</td>
<td>[1.648; 1.651] $\cdot 10^{-4}$</td>
<td>[1.621; 1.657] $\cdot 10^{-4}$</td>
</tr>
</tbody>
</table>
Overall results: State space

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

FTPP-1-1, Time bound 1

CI width

FTPP-1-1, Time bound 0.01

CI width

100.000 traces

60 seconds
Preliminary results on time-bounded reachability

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

<table>
<thead>
<tr>
<th>Time bound</th>
<th>Forcing /conditioning</th>
<th>CI width</th>
<th>Simulation time</th>
<th>CI width</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>None</td>
<td>$1.7 \cdot 10^{-10}$</td>
<td>0.16</td>
<td>$4.8 \cdot 10^{-12}$</td>
</tr>
<tr>
<td>1</td>
<td>Accel. $\times$ 2</td>
<td>$1.9 \cdot 10^{-10}$</td>
<td>0.15</td>
<td>$6.4 \cdot 10^{-12}$</td>
</tr>
<tr>
<td>1</td>
<td>Accel. $\times$ 15</td>
<td>$1.5 \cdot 10^{-9}$</td>
<td>0.15</td>
<td>$5.0 \cdot 10^{-11}$</td>
</tr>
<tr>
<td>1</td>
<td>Forcing</td>
<td>$1.4 \cdot 10^{-10}$</td>
<td>0.60</td>
<td>$1.2 \cdot 10^{-11}$</td>
</tr>
<tr>
<td>1</td>
<td>Conditioning</td>
<td>$1.0 \cdot 10^{-10}$</td>
<td>70</td>
<td>$1.3 \cdot 10^{-10}$</td>
</tr>
<tr>
<td>0.01</td>
<td>None</td>
<td>$2.1 \cdot 10^{-12}$</td>
<td>0.05</td>
<td>$3.1 \cdot 10^{-14}$</td>
</tr>
<tr>
<td>0.01</td>
<td>Accel. $\times$ 2</td>
<td>$9.8 \cdot 10^{-13}$</td>
<td>0.08</td>
<td>$1.7 \cdot 10^{-14}$</td>
</tr>
<tr>
<td>0.01</td>
<td>Accel. $\times$ 15</td>
<td>$1.5 \cdot 10^{-13}$</td>
<td>0.07</td>
<td>$2.9 \cdot 10^{-15}$</td>
</tr>
<tr>
<td>0.01</td>
<td>Forcing</td>
<td>$1.9 \cdot 10^{-13}$</td>
<td>0.36</td>
<td>$1.1 \cdot 10^{-14}$</td>
</tr>
<tr>
<td>0.01</td>
<td>Conditioning</td>
<td>$3.9 \cdot 10^{-14}$</td>
<td>53</td>
<td>$3.6 \cdot 10^{-14}$</td>
</tr>
</tbody>
</table>
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.
- Support for non-Markovian timing of e.g. maintenance actions.
- Limited nondeterminism to cover full set of DFTs.
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?