Counterexamples for Timed Probabilistic Reachability

Stefan Leue

University of Konstanz
Chair for Software Engineering

Stefan.Leue@uni-konstanz.de
http://www.inf.uni-konstanz.de/~soft

17 August 2005
SUNY Stony Brook

software engineering

Joint work with...

- **Husain Aljazzar**
  - University of Konstanz
- **Holger Hermanns**
  - Saarland University

Outline

- Motivation
- Probability Measures for Optimizing Search
- Approximation based on Uniformisation
- Directed Probabilistic Reachability Analysis
- Case Study
- Conclusion and Outlook

Outline

- Motivation
- Probability Measures for Optimizing Search
- Approximation based on Uniformisation
- Directed Probabilistic Reachability Analysis
- Case Study
- Conclusion and Outlook

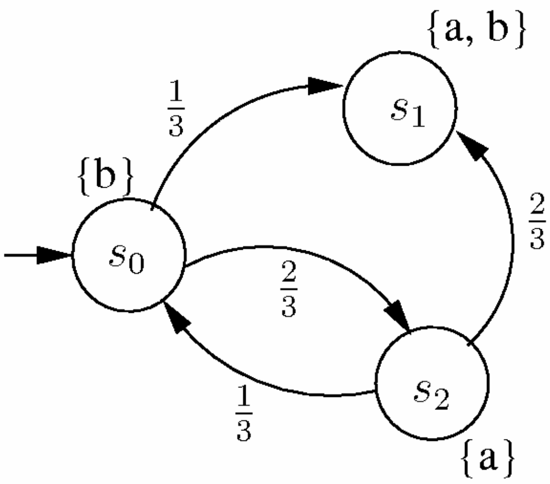
Motivation

- **Why Stochastic Model Checking?**
  - Stochastic models are widely used to model and analyze system performance and dependability.
    - communication protocols, embedded systems, etc.
  - Various model checking approaches for stochastic models have been presented.
  - Our point of reference: CSL Model checking
    - Baier, C., Haverkort, B., Hermanns, H., Katoen, J.P.: Model-Checking Algorithms for Continuous-Time Markov chains. IEEE Transitions on Software Engineering 29, 2003
      - Continuous Stochastic Logic (CSL) for expressing real-time probabilistic properties of Continuous Time Markov Chains (CTMCs) has been proposed.
        - Probabilitstic, timed extension of CTL.
      - Efficient approximative algorithms to model check CSL formulae have been developed (e.g., by the above authors).

Markov Chain Models

Discrete Time

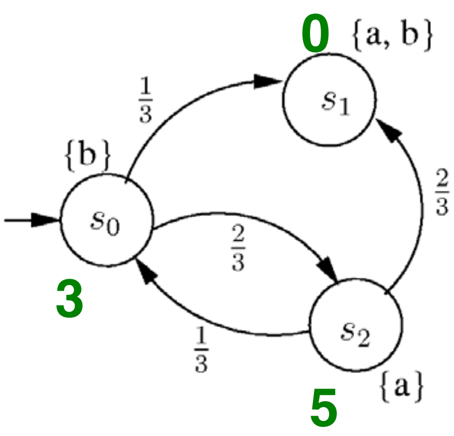
- A discrete time Markov chain (**DTMC**) is a quadruple (S, s_0, P, L), where
  - S is a finite set of states, and
  - $s_0 \in S$ is an initial state
  - $P : S \times S \rightarrow \mathbb{R}$ is a probability matrix, satisfying that for each state, the sum of the probabilities of outgoing probabilistic transitions is 1.
  - $L : S \rightarrow 2^{AP}$ is labeling function, which assigns each state the subset of valid atomic propositions.
  - i.e., a Kripke structure augmented with probabilistic information



Markov Chain Models

Continuous Time

- A continuous time Markov chain (**CTMC**) is a quintuple (S, s_0, P, E, L), where
  - (S, s_0, P, L) is a DTMC and
  - E : S $\rightarrow \mathbb{R}_{>0}$ is a function assigning each state an exit rate,
    - e.g., E := {(s_0, **3**), (s_1, **0**), (s_2, **5**)}
  - exit rates are exponentially distributed



Probabilities in DTMCs and CTMCs

- steady-state probabilities:
  - system is considered "in the long run", i.e., when equilibrum has been reached
- transient-state probabilities:
  - system is considered at a given time instant t

software engineering

Property Specification

- **Timed Pobabilistic Reachability**
  - The probability to reach a state s violating a state proposition ϑ, i.e., satisfying $\varphi := \neg \vartheta$, within the time interval [0, t], does not exceed a probability $p \in [0, 1]$.
  - Specification using Continuous Stochastic Logic (CSL)

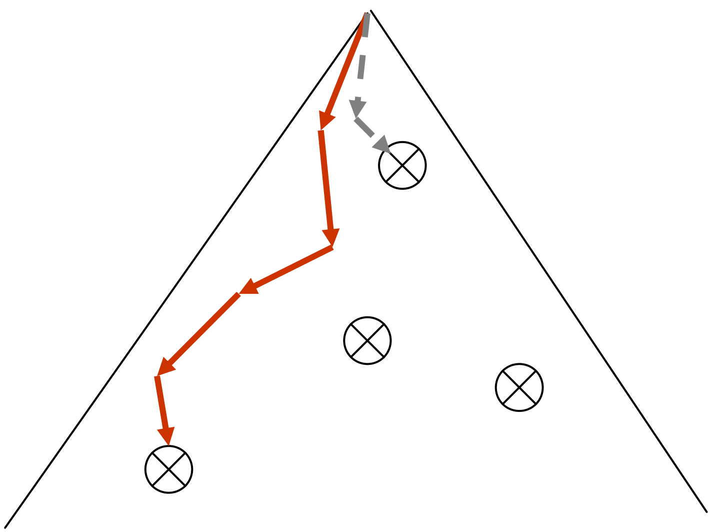
$$\mathcal{P}_{<p}(\Diamond^{\leq t}\varphi)$$

$\mathcal{P}_{<p}$: Transient probability does not exceed p.
$\Diamond^{\leq t}$: Timed reachability within [0, t]

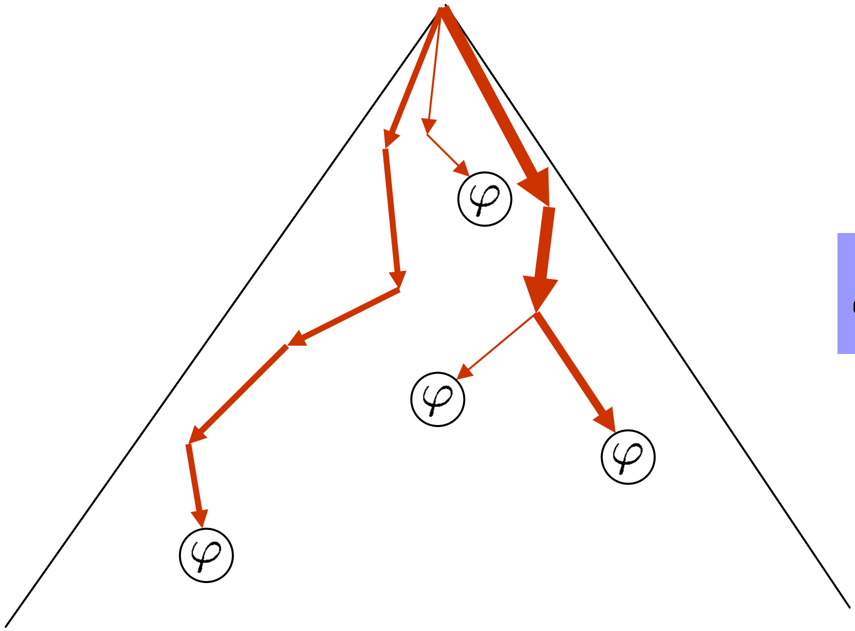
- **CSL Model Checking (according to Baier et al.)**
  - recursively determines sets of states satisfying CSL subformulae
  - efficient and numerically stable
  - based on uniformisation
  - Weakness:
    - CSL model checking (like many other stochastic model checking approaches) do not return "counterexamples"
      - problematic for system debugging
- **Approach**
  - state space search on the CTMC to find offending system runs

Explicit-State Model Checking

- **Explicit-State model checking (ESMC)**
  - checks state properties by exploring the state space using graph search algorithms like DFS and BFS.
  - If an error is found, an offending system run is returned, which helps in explaining why the property is violated.
- **What constitutes a good counterexample?**
  - In typical non-stochastic transition systems:
    - good = short
- **How to obtain good (short) counterexamples?**
  - Breadth-First Search (BFS).
  - Directed Explicit-State Model Checking (DESMC), uses heuristics guided search (e.g., Greedy Best-First or A*).



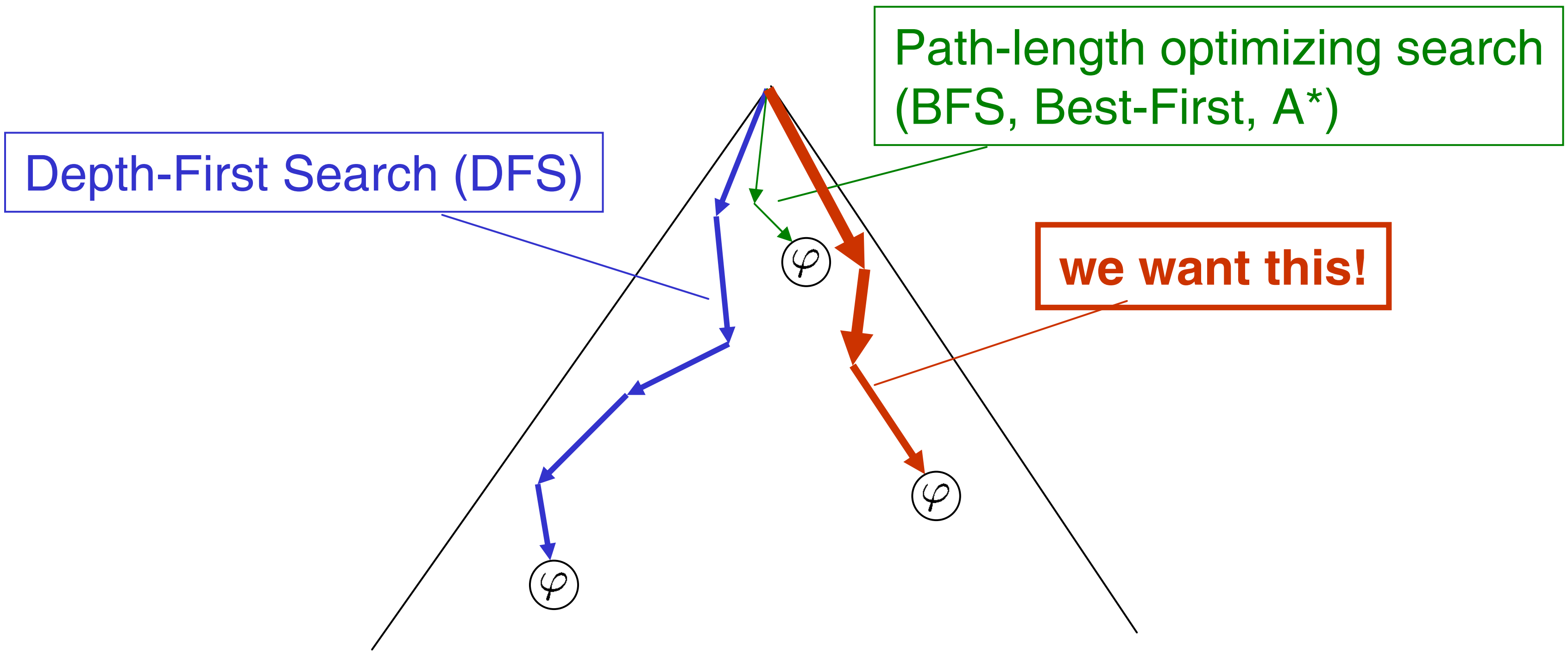
Probabilistic Timed Reachability



$$\phi := \mathcal{P}_{<p}(\Diamond^{\leq t}\varphi)$$

- **Property Violation**
  - According to CSL semantics, validity of ϕ can be decided by comparing the probability bound p with cumulated reachability probability $\sum_{s' \models \varphi} \rho(s', s_{init}, t)$.
    - probability measure of the (tree-shaped) infinite cylinder set containing all paths that reach φ-state within t time units
    - can be computed by transient analysis where all φ-states are made absorbing (CSL model checking à la Baier et al.)

- **Search Algorithms**
  - What do standard state space search algorithms deliver when applied to stochastic models?



  - Need search algorithms that optimize (maximize) probability mass along single paths.

Outline

- Motivation
- **Probability Measures for Optimizing Search**
- Approximation based on Uniformisation
- Directed Probabilistic Reachability Analysis
- Case Study
- Conclusion and Outlook

Paths and Runs

- **DTMC (S, s_0, P, L)**
  - An infinite run is a sequence

    $$s_0 \longrightarrow s_1 \longrightarrow s_2 \longrightarrow \ldots,$$

    with $(\forall i>0)(\mathcal{P}(s_i, s_{i+1}) > 0)$
  - A finite run is a sequence

    $$s_0 \longrightarrow s_1 \longrightarrow s_2 \longrightarrow \ldots \longrightarrow s_n,$$

    with $\forall$ i ($0 \leq i < n$): $\mathcal{P}(s_i, s_{i+1}) > 0$ and s_n is absorbing.
    - An absorbing state (of a DTMC) is a state which has only self transitions as outgoing transitions.
- **CTMC (S, s_0, P, E, L)**
  - An infinite path is a sequence $s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} s_2 \xrightarrow{t_2} \ldots,$ where $s_0 \longrightarrow s_1 \longrightarrow s_2 \longrightarrow \ldots$ is an infinite run in the DTMC (S, s_0, P, L).
  - A finite path is a sequence $s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} s_2 \xrightarrow{t_2} \ldots \xrightarrow{t_{n-1}} s_n,$ where $s_0 \longrightarrow s_1 \longrightarrow s_2 \longrightarrow \ldots \longrightarrow s_n$ is a finite run in the DTMC (S, s_0, P, L).
  - Note: each run yields an infinite set of paths!

Timed Reachability Probability

- **The Timed Reachability Probability *ρ(s, s‘, t)* of a CTMC**
  - probability to reach state s' at the latest at time t, if starting in state s at time 0.

$$\rho(s', s, t) := Pr\{\sigma \in Path(s) \mid \exists t' \in [0, t] : \sigma @ t' = s'\}$$

    - Pr is the probability mass of the above set.
    - Path(s) is the set of paths starting at the state s.
    - σ@t‘ is the state occupied by the system at the time point t', if the path σ is executed.
  - computation of ρ(s, s‘, t) can be reduced to time-dependent state probability

    $\pi(s', s, t) = Pr\{\sigma \in Path(s) \mid \sigma @ t = s'\}$

    after making s' absorbing
    - determines probability to reach state s' at time t when starting in s at time 0
    - efficient uniformisation based techniques to compute this exist

software engineering

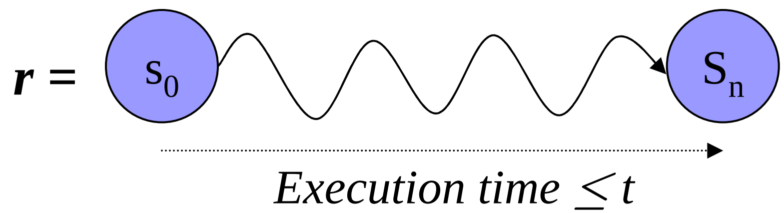
Counterexamples for Stochastic Models

- **What is a Good Counterexample in Stochastic Models?**
  - The violation of a timed probabilistic reachability property in a CTMC caused not only by one run, but by an infinite set of runs from a tree of unbounded depth.
    - Infinite branching tree due to varying real-time stamps.
  - We expect the user to be interested in a counterexample which carries a high probability mass (i.e., is most informative).
    - Helps identify the portion of the infinite set of runs that violate probability bound which is undesired.
  - The length of a path is not indicative of its probability mass.
    - → BFS or (D)ESMC with the length as a guiding cost measure will not help!
  - We aim to select an offending system run whose contribution to the timed reachability probability is high or even maximal.
    - → timed run probability

Timed Run Probability for CTMCs

- Let r = $s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \ldots \rightarrow s_n$ a finite run of a CTMC.
- The timed run probability of r, $\gamma(r, t)$, is the probability to execute run r within the time interval [0, t]:

$$\gamma(r,t) = \Pr\{\sigma \in Path(s_0) \mid \exists t' \in [0,t] : \sigma@t' = s_n \wedge \sigma{\downarrow}_{s_n} = r\}.$$



- Intuitively, $\gamma(r, t)$ gives the probability that r is executed and s_n=last(r) is reached at the latest at time t.
- For finite run r, γ is given by

$$\gamma(r,t) = \int_0^t \left(p(s_1, s_0, t_1) \cdot \left(\ldots \left(\int_0^{t - t_{n-1}} p(s_n, s_{n-1}, t_n) \cdot dt_n \right) \ldots \right) \right) \cdot dt_1,$$

  where $p(s', s, t) = P(s,s') \cdot (1 - e^{-E(s) \cdot t})$ is the probability to move from s to s' in the interval [0,t].
- γ(r,t,) can be computed by ρ(last(r), first(r), t) on a CTMC for which all states not reached by r are made absorbing

Outline

- Motivation
- Probability Measures for Optimizing Search
- **Approximation based on Uniformisation**
- Directed Probabilistic Reachability Analysis
- Case Study
- Conclusion and Outlook

Optimizing ESMC for Stochastic Models

- **Idea**
  - Use of an optimizing state space search algorithm with the timed run probability as optimization criterion!
  - In each search iteration we have to compute the timed run probability for the runs from the initial state to each newly explored state.
    - This needs to be done on-the-fly!
  - However, the determination of the exact value of γ (r, t) is computationally very expensive.
    - Requires solving complicated nested integral.
      - Computationally expensive and prone to numerical instability problems.
    - This cannot be done on-the-fly!
  - A powerful approximation is required!

Approximation based on the uniformised model!

- **Uniformisation for a CTMC:**
  - Uniformise A into a DTMC A' for which a timed run probability γ' can easily be computed:
    - Let A=(S, P, E, L) a CTMC.
    - Choose a number Γ with $\Gamma \geq$ E(s) for all s $\in$ S.
    - The transition probability matrix M for DTMC A'=(S, M, L) is defined as follows:

$$M = I + \frac{1}{\Gamma} \cdot E(s) \cdot (P - I)$$

where I is the identity matrix.

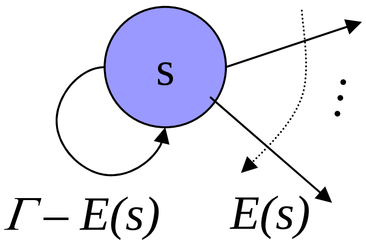
Uniformisation

- **Uniformisation for a CTMC:**
  - A' is then embedded into a Poisson process as follows:

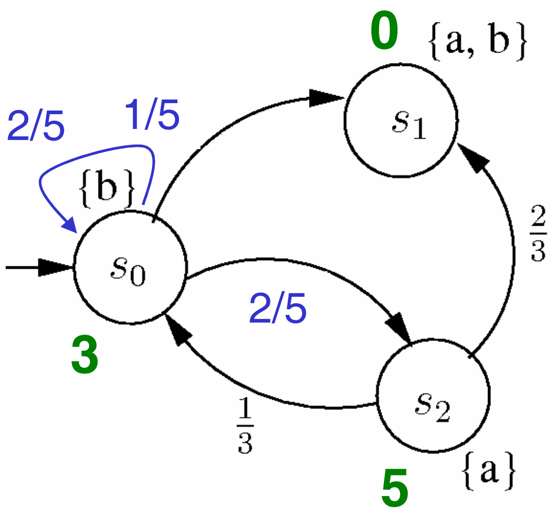
$$Prob\{N(t) = k\} := \frac{(\Gamma \cdot t)^k}{k!} \cdot e^{-\Gamma \cdot t}, \qquad k,\ t \geq 0.$$

  - Expected value is N := $\Gamma \cdot$ t.
    - N corrsponds to number of hops in A' that may occur in t time units.
    - Probability of N hops in t time units is maximal
- **Use in State Space Search (Now on A')**
  - t: time bound in property
  - search selects path in A' with length of at most N transitions, i.e. that carries maximal probability
    - limit search to states probably reachable within [0, t]

- **Intuitively, what does this mean?**
  - For each state s, the exit rate E(s) is increased to be Γ.
  - A self loop carrying the difference between E(s) and Γ is added to s.
  - The model performs discretely, i.e. on each event exactly one transition is fired.



- **In the Example**
  - Let Γ = 5



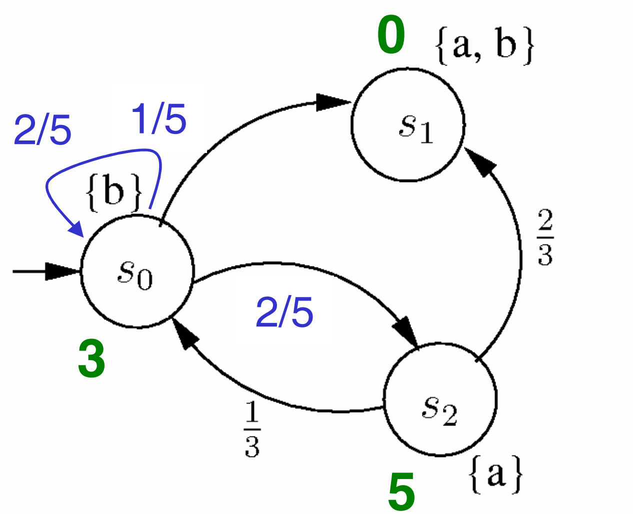
CTMC Timed Run Probability Approximation

- γ(r, t) (in A) is approximated by DTMC timed run probability **γ'(r, N)** (in A'=(S, P', L)).
  - γ'(r, N): reachability property in A' along r bounded by N hops
  - traversal tree of search algorithm has always at most one run r between each pair of states, i.e., run is characterized by (first(r), last(r)) and we write γ'(last(r), first(r), N) or γ'(last(r), N).
  - π' denotes restriction of π to the traversal tree
    - π'(s, k) is $\pi(s, s_{init}, k)$ on DTMC obtained from A' by redirecting all transitions not contained in traversal tree, with the exception of self-loops, to an absorbing state.
  - γ'(r, N) can easily be computed by

$$\gamma'(s, N) = M(pred(s), s) \cdot \sum_{k=0}^{N-1} \pi'(pred(s), k)$$

- **Computing γ'**

$$\gamma'(s,N) = M(pred(s),s) \cdot \sum_{k=0}^{N-1} \pi'(pred(s),k)$$

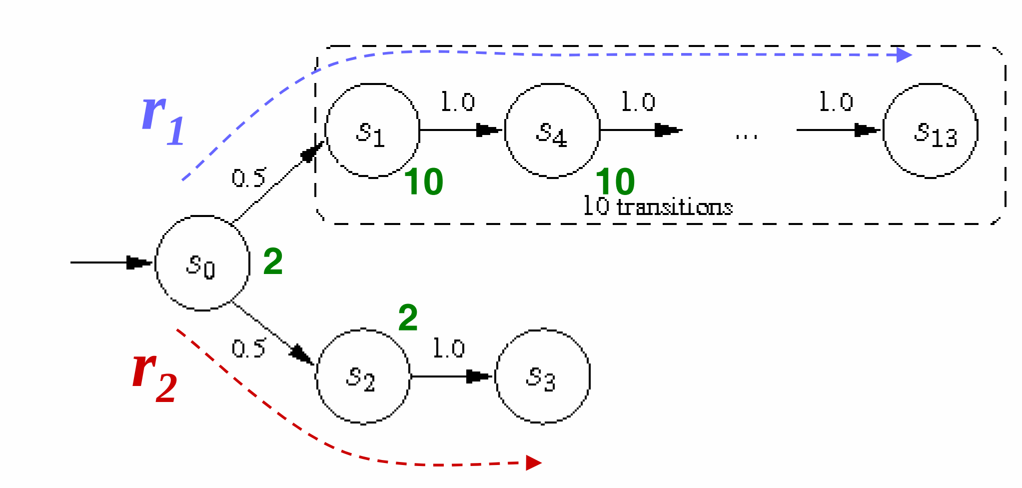


Let $r_1 = s_0 \rightarrow s_1$ and $r_2 = s_0 \rightarrow s_2 \rightarrow s_1$, then:

$$\gamma'(r_1,2) = \tfrac{1}{5} \cdot (\pi_{r_1}(s_0,0) + \pi_{r_1}(s_0,1)) = \tfrac{1}{5} \cdot (1 + 1 \cdot \tfrac{2}{5}) = \tfrac{7}{25}$$

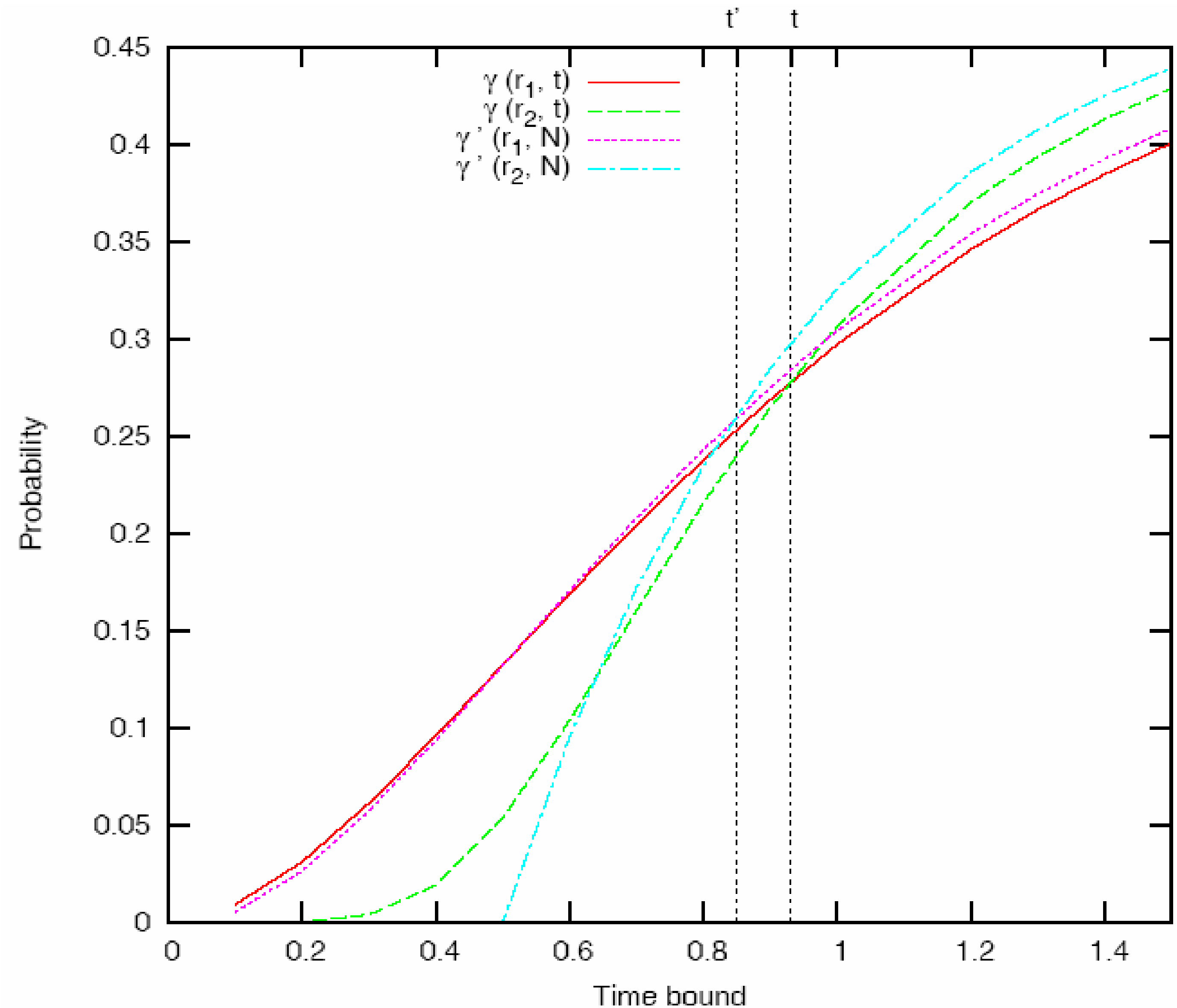
$$\begin{aligned}\gamma'(r_2,2) &= \tfrac{2}{3} \cdot (\pi_{r_2}(s_1,0) + \pi_{r_2}(s_1,1)) \\ &= \tfrac{2}{3} \cdot (0 + \tfrac{2}{5} \cdot \pi_{r_2}(s_0,0)) = \tfrac{2}{3} \cdot (0 + \tfrac{2}{5} \cdot 1) = \tfrac{4}{15}\end{aligned}$$

♦ An Intriguing Example.



- Note: the path with optimal run time probability changes with the time bound t!
  - For $t < 1.0$, $\gamma_{CT}(r_1, t) > \gamma_{CT}(r_2, t)$
  - For $t > 1.0$, $\gamma_{CT}(r_1, t) < \gamma_{CT}(r_2, t)$

Quality of Uniformisation Approximation



software engineering

Outline

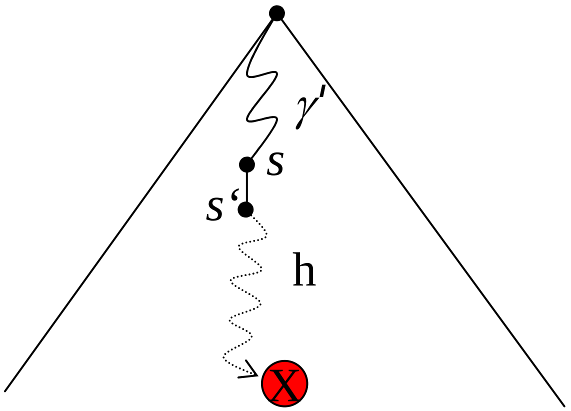
- Motivation
- Probability Measures for Optimizing Search
- Approximation based on Uniformisation
- **Directed Probabilistic Reachability Analysis**
- Case Study
- Conclusion and Outlook

Directed Probabilistic Reachability Analysis

- **We are now able to**
  - explore CTMCs (and DTMCs) using optimizing algorithms, and
  - select runs (counterexamples) which are approximating the optimal objective function (timed run probability) values.
- **Informed, Heuristics-guided Search Algorithms**
  - Use knowledge about the structural properties of the state space or the goal state specification to perform heuristics guided state space exploration.
    - Greedy Best First Search (GBestFS) and
    - Z*
      - generalization of A*, allows the use of non-additive cost measures
  - Such knowledge manifests itself in the heuristic evaluation function f which estimates the desirability of expanding a state.
  - f is based on intuition expressed through a heuristic function h, amongst others.

Directed Search Algorithms

- **Expansion of some state s:**
  - Generate the successor s‘
  - Compute f(s’) = F[ψ(s), f(s), h(s‘)]

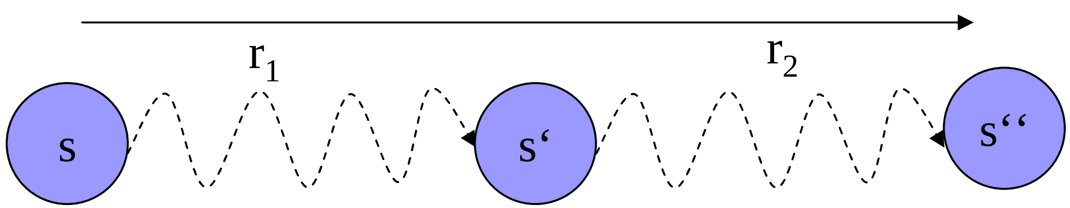


- **GBestFS: f(s‘) = h(s‘)**

- **Z*:**

$$\begin{aligned} f(s') &:= F[\psi(s), f(s), h(s')] \\ &:= F[\{\pi(s,k) | 0 \leq k \leq N\}, M(s,s'), h(s')] \\ &:= -\gamma'(s',N) \cdot h(s'). \end{aligned}$$

  - We conjecture that this cost measure delivers optimal solutions for the approximated model:



$$\gamma'(r,N) \leq \gamma'(r_1,N) \cdot \gamma'(r_2,N)$$

Determining and Computing Heuristic Functions

- Admissibility / informativeness of heuristics
  - admissibility: heuristic function h is optimistic and overestimates the maximal timed run probability until a state satisfying φ is reached outgoing from s.
    - desirable, but optimal solution is not the penultimate goal
  - informedness: heuristics discriminates well between desirable and undesirable states to be explored
    - desirable, since it reduces search effort
- If φ is an atomic state proposition, the construction of h depends on the domain and φ itself.
  - For complicated formulae involving Boolean connectives we suggest computing heuristics as illustrated in the following table:

φ	h_φ	$\bar{h}_\varphi$
$\neg\varphi_1$	$\bar{h}_{\varphi_1}$	h_{φ_1}
$\varphi_1 \vee \varphi_2$	$max\{h_{\varphi_1}, h_{\varphi_2}\}$	$min\{\bar{h}_{\varphi_1}, \bar{h}_{\varphi_2}\}$
$\varphi_1 \wedge \varphi_2$	$min\{h_{\varphi_1}, h_{\varphi_2}\}$	$max\{\bar{h}_{\varphi_1}, \bar{h}_{\varphi_2}\}$

e.g., for $\mathcal{P}_{<p}(\Diamond^{\leq t}(\varphi_1 \wedge \neg\varphi_2))$

Outline

- Motivation
- Probability Measures for Optimizing Search
- Approximation based on Uniformisation
- Directed Probabilistic Reachability Analysis
- **Case Study**
- Conclusion and Outlook

SCSI-2 Protocol

- Storage system consisting of up to 8 devices, one disk controller and up to 7 hard disks.
  - Assumption: one main disk, the remainder are backup disks.
  - Interested in probability to overload one of the disks.
- These devices are connected by a bus implementing the small computer system interface-2 (SCSI-2) standard.
- Each device is assigned a unique SCSI number between 0 and 7.
- The controller can send a command (CMD) to the disk d. After processing this command, the disk sends a reconnect message (REC) to the controller.
- CMD and REC messages of every disk are stored in two eight-place FIFO queues.
- CMD and REC messages circulate on the SCSI bus, which is shared by all devices.
- This system is modeled in LOTOS and transformed into an interactive Markov chain (IMC) by the CADP toolbox.

Property Specification

- **Properties**
  - to model disk load
    - φ_d: the command queue of disk d is full
    - ϑ_d: the command queue of disk d is empty
  - properties in CSL
    - MDOL: $\phi := \mathcal{P}_{<p}(\Diamond^{\leq t} \varphi_0 \wedge \vartheta_1 \wedge \vartheta_2)$
    - BDOL: $\theta := \mathcal{P}_{<p}(\Diamond^{\leq t} \vartheta_0 \wedge (\varphi_0 \wedge \vartheta_1) \vee (\vartheta_0 \wedge \varphi_1))$
- **Heuristics**
  - cq(s,i): for each disk i, number of commands contained in its command queue in state s
  - Markovian transitions
    - λ_d: delay required to issue new command to disk d
    - μ_d: servicing time of disk d
- **Uniformisation**
  - maximum exit rate: $max\{E(s) \mid s \in S\} = \sum_{d \in D} (\lambda_d + \mu_d) =: E_{max}$
    - replace any rate in model by rate/E_{max}

- **Optimisitc Heuristc Estimates**
  - heuristic functions (easy to compute)

$$h_{\varphi_d}(s) := (\frac{\lambda_d}{E_{max}} \cdot \sum_{k=0}^{N-1} (1 - p_{out}(s))^k))^{8-cq(s,d)}$$

$$h_{\vartheta_d}(s) := (\frac{\mu_d}{E_{max}} \cdot \sum_{k=0}^{N-1} (1 - p_{out}(s))^k))^{cq(s,d)}$$

where $p_{out}(s)$ is the branching probability of leaving s

  - conjectures establishing optimality in the approximated model

$$h_{\varphi_d}(s) \geq h^*_{\varphi_d}(s) := max\{\gamma'(s, s', N) \mid cq(s', d) = 8\}$$
$$h_{\vartheta_d}(s) \geq h^*_{\vartheta_d}(s) := max\{\gamma'(s, s', N) \mid cq(s', d) = 0\}$$

Case-Study: SCSI-2 Protocol

Experimental Results: Probabilities

	Time bound		1	2	3	4	5	6	7	8	9	10
MDOL	Model		0.235	0.312	0.327	0.329	0.329	0.329	0.330	0.330	0.330	0.330
	DFS		-	-	-	-	-	-	0.000	-	-	0.000
	BFS		-	0.161	0.161	0.161	0.161	0.161	0.161	0.161	0.161	0.161
	Dijkstra	estimated	-	0.049	0.049	0.049	0.049	0.049	0.049	0.049	0.049	0.049
		precise	-	0.161	0.161	0.161	0.161	0.161	0.161	0.161	0.161	0.161
	GBestFS	estimated	-	0.005	0.005	0.005	0.005	0.005	0.005	0.005	0.005	0.005
		precise	-	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012
	Z*	estimated	-	0.049	0.049	0.049	0.049	0.049	0.049	0.049	0.049	0.049
		precise	-	0.161	0.161	0.161	0.161	0.161	0.161	0.161	0.161	0.161

- Model:
  - total reachability property, as determined by numerical transient probability analyzer in CADP
- DFS:
  - either finds no counterexample within depth bound, or
  - finds counterexample with very low probability mass
- BFS:
  - probability mass of step-length optimal counterexample
  - happens to be the probability-mass optimal counterexample

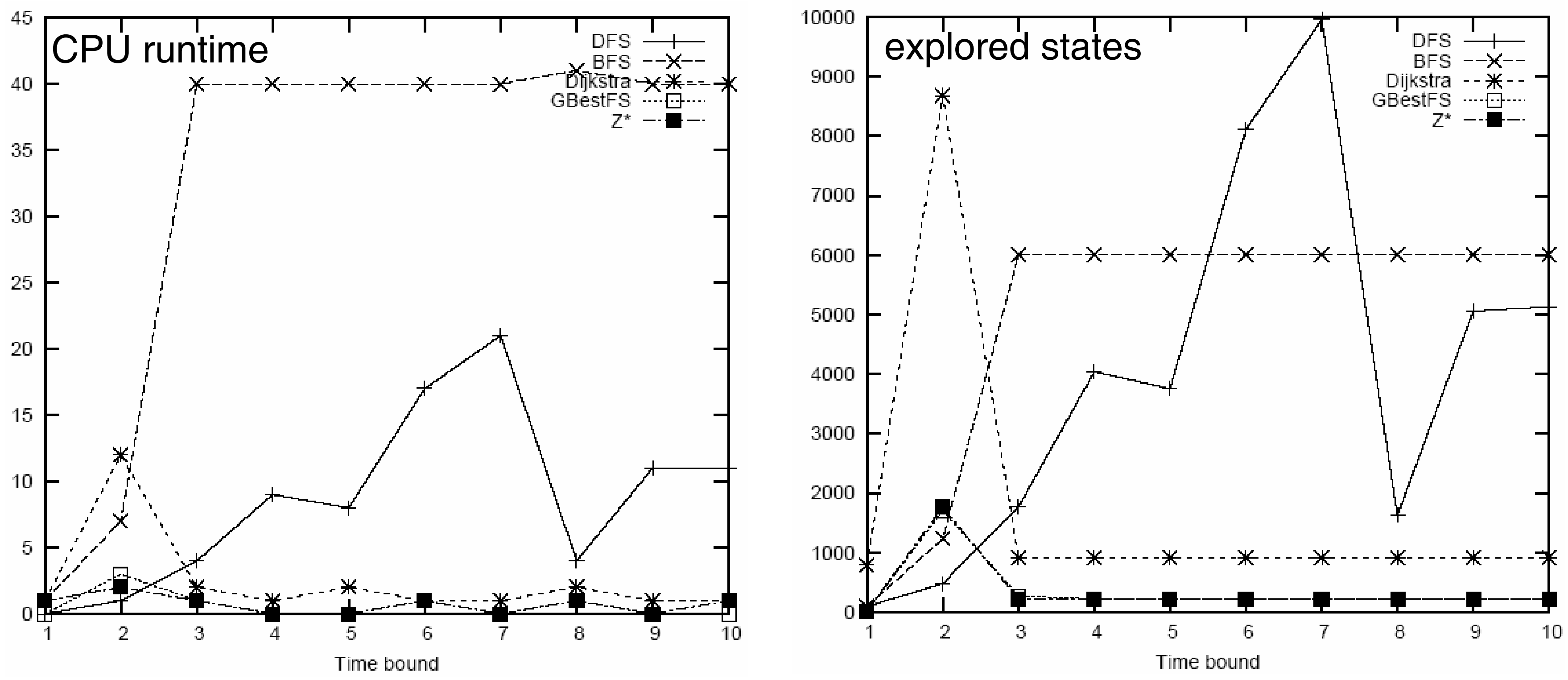
Case-Study: SCSI-2 Protocol

♦ Experimental Results: Probabilities

	Time bound		1	2	3	4	5	6	7	8	9	10
MDOL	Model		0.235	0.312	0.327	0.329	0.329	0.329	0.330	0.330	0.330	0.330
	DFS		-	-	-	-	-	-	0.000	-	-	0.000
	BFS		-	0.161	0.161	0.161	0.161	0.161	0.161	0.161	0.161	0.161
	Dijkstra	estimated	-	0.049	0.049	0.049	0.049	0.049	0.049	0.049	0.049	0.049
		precise	-	0.161	0.161	0.161	0.161	0.161	0.161	0.161	0.161	0.161
	GBestFS	estimated	-	0.005	0.005	0.005	0.005	0.005	0.005	0.005	0.005	0.005
		precise	-	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.012
	Z*	estimated	-	0.049	0.049	0.049	0.049	0.049	0.049	0.049	0.049	0.049
		precise	-	0.161	0.161	0.161	0.161	0.161	0.161	0.161	0.161	0.161

- Dijkstra (uses $-\gamma'(s,N)$ as weights):
  - delivers optimal estimated model
  - high precise probability $\gamma(r, t)$ in original model
- GBestFS (informed, uses approximation based heuristics)
  - finds a low probability counterexample both in approximated model (estimate) and in the original model (precise)
- Z* (informed, uses approximation based heuristics)
  - finds same counterexamples as Dijkstra, which supports our claim of optimality in the approximated model.

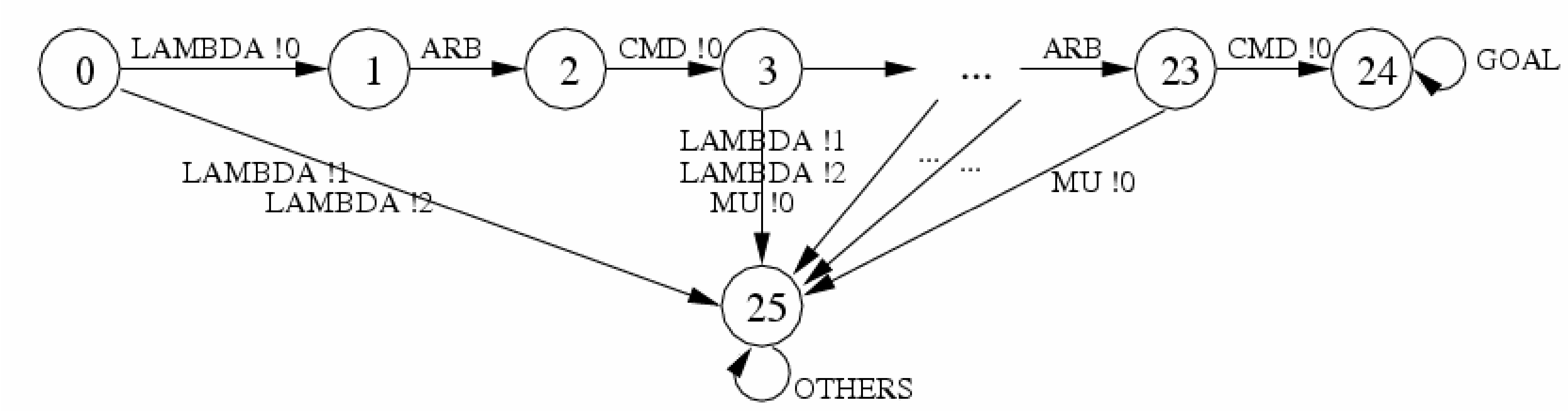
- **Experimental Results: MDOL, computational effort**



  - informed algorithms (GBestFS, Z*) better performance than uninformed algorithms (DFS, BFS, Dijkstra)
  - sometimes, GBestFS shows a slightly a better performance than Z*

Experimental Results: Qualitative Analysis

- DFS finds goal state on a very intricate run that carry very little probability mass
- Z* finds a counterexample, that quite intuitively carries high probability
  - right from the start, the disk continually receives commands without getting a chance to service them
    - LAMBDA !0: Markovian delay (relatively high compared to other Markovian delays in the system)
    - ARB: access to data bus
    - CMD !0: command to disk 0.



Outline

- Motivation
- Probability Measures for Optimizing Search
- Approximation based on Uniformisation
- Directed Probabilistic Reachability Analysis
- Case Study
- **Conclusion and Outlook**

Conclusion

- **Counterexamples**
  - defined counterexamples for CTMCs, including their probability mass: timed run probabilities
  - approximate the computationally expensive computation of timed run probabilities through uniformisation
- **Directed CTMC Exploration**
  - use approximative timed run probability in determining generating path costs
  - combine with domain specific information to compute admissible heuristic estimates (admissible in the approximated model)
- **Experimental Evaluation (SCSI-2)**
  - using approximated timed run probabilities allows Dijkstra and heuristic search algorithms to find meaningful counterexamples
  - heuristics guided search is computationally superior to uninformed search

Outlook

- **Threats to Valitidy**
  - more experimental data
    - convergence to PRISM tool environment, more models available
    - use randomly generated models
- **Underapproximation of Probabilistic Timed Reachability**
  - find tree of offending system runs so that combined probability mass exceeds probability bound
  - potentially computationally much more efficient that precise solution of problem
- **Application to Other Stochastic Models**
  - Continuous Time Markov Decision Processes
    - contain non-determinism

Reference

- H. Aljazzar, H. Hermanns and S. Leue. *Counterexamples in Timed Probabilistic Reachability*. To appear in: Proceedings of FORMATS'05, Lecture Notes in Computer Science, Springer Verlag, 2005.

software engineering

Overflow

- **DTMC**
  - $\pi(s',s,k) = P^k(s,s')$