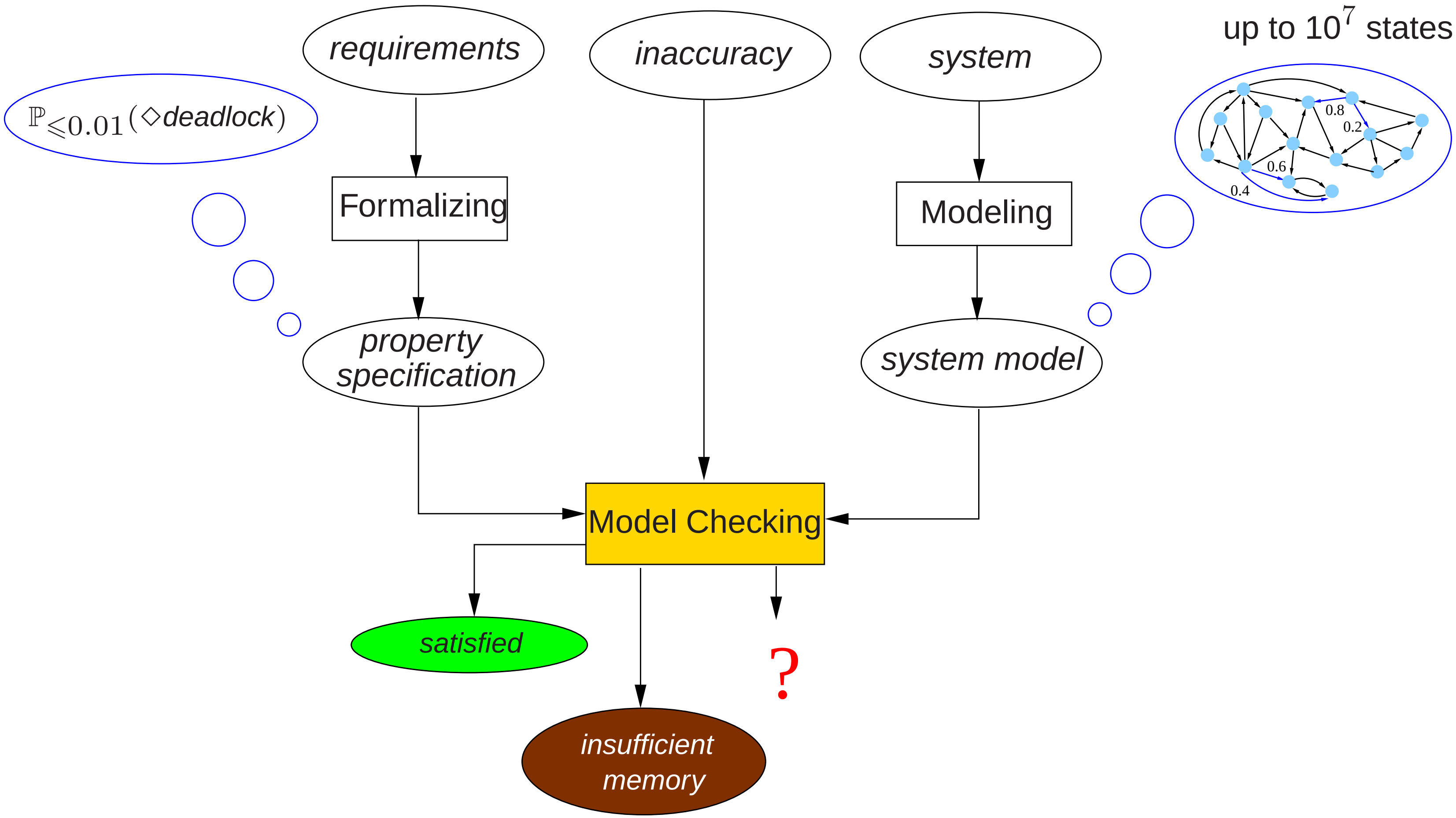
Regular Expressions for PCTL Counterexamples

Berteun Damman, Tingting Han, and Joost-Pieter Katoen

Software Modeling and Verification, RWTH Aachen University
and Formal Methods and Tools, University of Twente

QEST'08, September 16, Saint Malo

Probabilistic model checking



Counterexamples

- Are of utmost importance:
  - diagnostic feedback, key to abstraction-refinement, schedule synthesis . . .
  - fit to paradigm “model checking = bug hunting”

Counterexamples

- Are of utmost importance:
  - diagnostic feedback, key to abstraction-refinement, schedule synthesis . . .
  - fit to paradigm “model checking = bug hunting”
- LTL counterexamples are finite paths
  - $\Box \Phi$: a path ending in a $\neg \Phi$-state
  - $\Diamond \Phi$: a $\neg \Phi$-path leading to a $\neg \Phi$ cycle
  - BFS yields shortest counterexamples

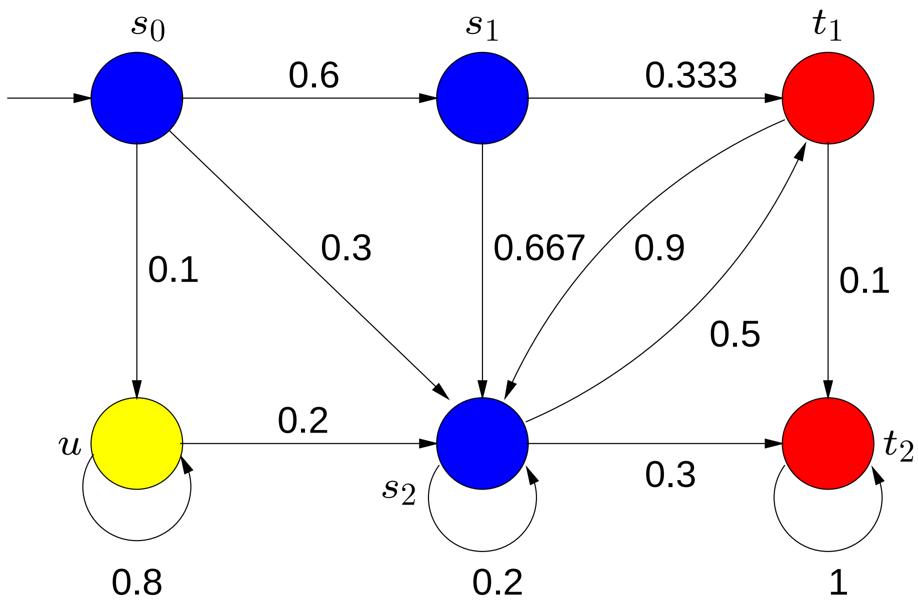
Counterexamples

- Are of utmost importance:
  - diagnostic feedback, key to abstraction-refinement, schedule synthesis . . .
  - fit to paradigm “model checking = bug hunting”
- LTL counterexamples are finite paths
  - $\Box\Phi$: a path ending in a $\neg\Phi$-state
  - $\Diamond\Phi$: a $\neg\Phi$-path leading to a $\neg\Phi$ cycle
  - BFS yields shortest counterexamples
- CTL counterexamples are (mostly) finite trees
  - universal CTL$\setminus$LTL: trees or proof-like counterexample
  - existential CTL: witnesses, annotated counterexample

Counterexamples

- Are of utmost importance:
  - diagnostic feedback, key to abstraction-refinement, schedule synthesis . . .
  - fit to paradigm "model checking = bug hunting"
- LTL counterexamples are finite paths
  - $\Box\Phi$: a path ending in a $\neg\Phi$-state
  - $\Diamond\Phi$: a $\neg\Phi$-path leading to a $\neg\Phi$ cycle
  - BFS yields shortest counterexamples
- CTL counterexamples are (mostly) finite trees
  - universal CTL$\setminus$LTL: trees or proof-like counterexample
  - existential CTL: witnesses, annotated counterexample
- This talk: PCTL counterexamples for DTMCs

Discrete-time Markov Chain



a DTMC is a triple $(S, \mathbf{P}, L)$ with state space S and state-labelling L

and $\mathbf{P}$ a stochastic matrix with $\mathbf{P}(s, s') =$ one-step probability to jump from s to s'

Probabilistic CTL (Hansson & Jonsson, 1994)

- For $a \in AP$, $J \subseteq [0,1]$ an interval with rational bounds, and $h \in \mathbb{N}$:

$$\Phi \; ::= \; a \;\Big|\; \Phi \wedge \Phi \;\Big|\; \neg\Phi \;\Big|\; \mathbb{P}_J(\varphi)$$

$$\varphi \; ::= \; \Phi \,\mathsf{U}\, \Phi \;\Big|\; \Phi \,\mathsf{U}^{\leqslant h}\, \Phi$$

- $s_0 s_1 s_2 \ldots \models \Phi \,\mathsf{U}^{\leqslant h}\, \Psi$ if Φ holds until Ψ holds within h steps

- $s \models \mathbb{P}_J(\varphi)$ if probability of set of φ-paths starting in s lies in J

abbreviate $\mathbb{P}_{[0,0.5]}(\varphi)$ by $\mathbb{P}_{\leqslant 0.5}(\varphi)$ and $\mathbb{P}_{]0,1]}(\varphi)$ by $\mathbb{P}_{>0}(\varphi)$ and so on

This talk

- What is a PCTL counterexample?
  - a set of paths with sufficient probability mass
- How to determine smallest counterexamples?
  - exploit k-shortest path algorithms
- How about the size of counterexamples?
  - well, they may be excessively large and incomprehensible
- Can we do better?
  - yes, represent counterexamples by regular expressions!
- How to obtain (short) regular expressions?
  - use automata theory and some heuristics

This talk

- What is a PCTL counterexample? [Han & Katoen, TACAS'07]]
  - a set of paths with sufficient probability mass
- How to determine smallest counterexamples?
  - exploit k-shortest path algorithms
- How about the size of counterexamples?
  - well, they may be excessively large and incomprehensible
- Can we do better?
  - yes, represent counterexamples by regular expressions!
- How to obtain (short) regular expressions?
  - use automata theory and some heuristics

This talk

- What is a PCTL counterexample?
  - a set of paths with sufficient probability mass
- How to determine smallest counterexamples?
  - exploit k-shortest path algorithms
- How about the size of counterexamples? [This QEST'08 paper]
  - well, they may be excessively large and incomprehensible
- Can we do better?
  - yes, represent counterexamples by regular expressions!
- How to obtain (short) regular expressions?
  - use automata theory and some heuristics

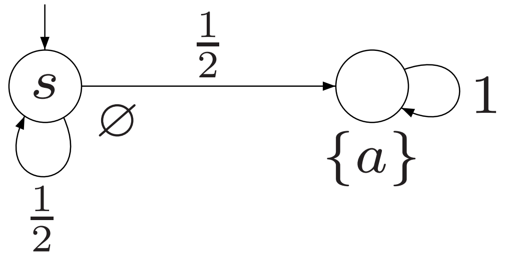
PCTL counterexamples for $s \not\models \mathbb{P}_{\leqslant p}(\varphi)$

- A *counterexample* C is a set of $\underbrace{\text{finite paths}}_{\text{evidences}}$ satisfying
  - $\sigma \in C$ implies σ starts in s and $\sigma \models \varphi$
  - $\Pr(C) = \sum_{\sigma \in C} \mathbf{P}(\sigma)$ exceeds p

PCTL counterexamples for $s \not\models \mathbb{P}_{\leqslant p}(\varphi)$

- A *counterexample* C is a set of $\underbrace{\text{finite paths}}_{\text{evidences}}$ satisfying
  - $\sigma \in C$ implies σ starts in s and $\sigma \models \varphi$
  - $\Pr(C) = \sum_{\sigma \in C} \mathbf{P}(\sigma)$ exceeds p

- Property: counterexamples for non-strict bounds $\leqslant p$ are *finite*



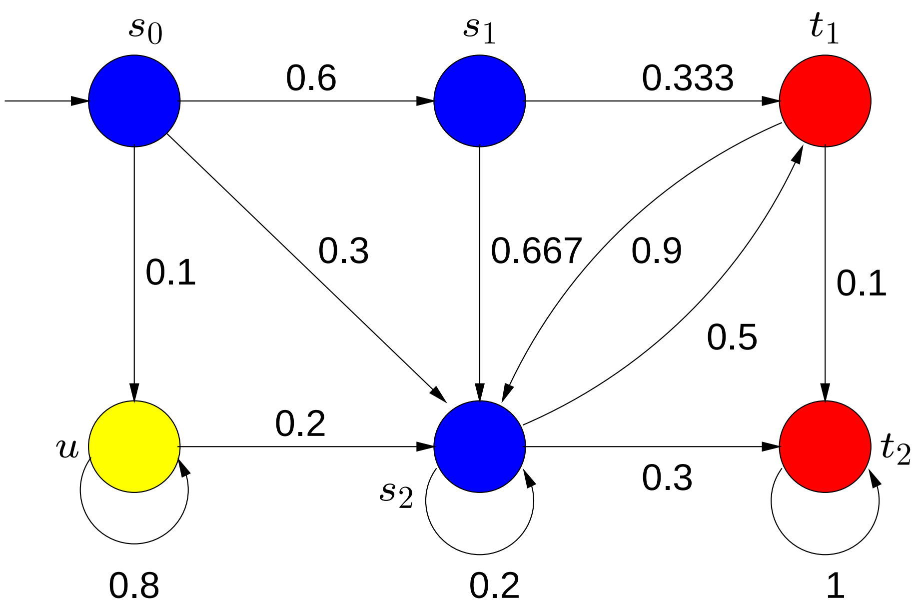
A DTMC with infinite counterexample for $s \not\models \mathbb{P}_{<1}(\Diamond\, a)$

PCTL counterexamples for $s \not\models \mathbb{P}_{\leqslant p}(\varphi)$

- A *counterexample* C is a set of finite paths (evidences) satisfying
  - $\sigma \in C$ implies σ starts in s and $\sigma \models \varphi$
  - $\Pr(C) = \sum_{\sigma \in C} \mathbf{P}(\sigma)$ exceeds p
- Property: counterexamples for non-strict bounds $\leqslant p$ are *finite*
- C is *minimal* if $|C| \leqslant |C'|$ for any counterexample C'
- C is *smallest* if:

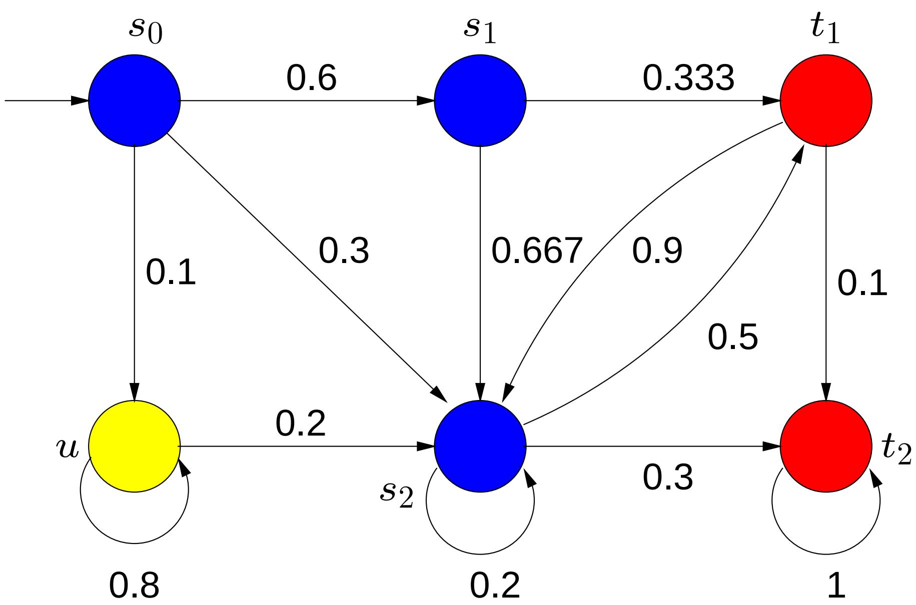
  C is minimal, and $\Pr(C) \geqslant \Pr(C')$ for any minimal counterexample C'

Evidences for $s_0 \not\models \mathbb{P}_{\leqslant \frac{1}{2}}(a \,\mathsf{U}\, b)$



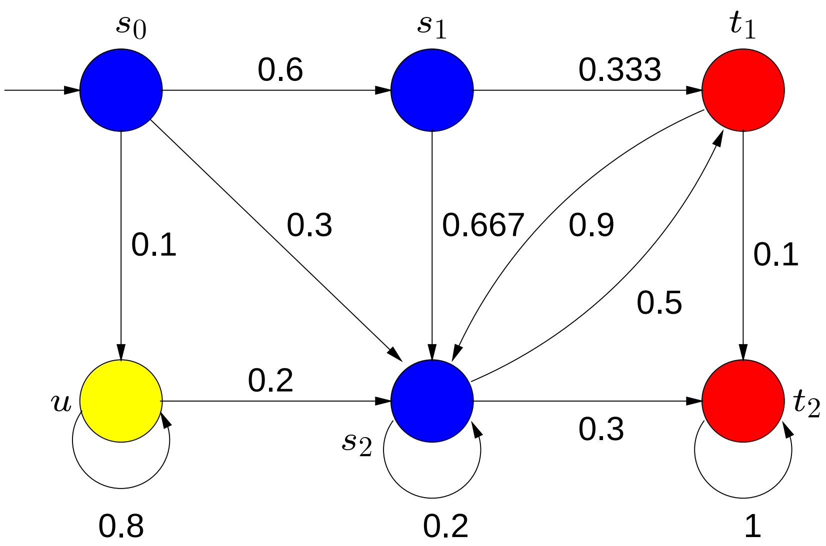
evidences	prob.
$\sigma_1 = s_0\, s_1\, t_1$	0.2
$\sigma_2 = s_0\, s_1\, s_2\, t_1$	0.2
$\sigma_3 = s_0\, s_2\, t_1$	0.15
$\sigma_4 = s_0\, s_1\, s_2\, t_2$	0.12
$\sigma_5 = s_0\, s_2\, t_2$	0.09
$\ldots$	$\ldots$

Strongest evidences (SEs)



evidences	prob.
$\sigma_1 = s_0\, s_1\, t_1$	0.2
$\sigma_2 = s_0\, s_1\, s_2\, t_1$	0.2
$\sigma_3 = s_0\, s_2\, t_1$	0.15
$\sigma_4 = s_0\, s_1\, s_2\, t_2$	0.12
$\sigma_5 = s_0\, s_2\, t_2$	0.09
...	...

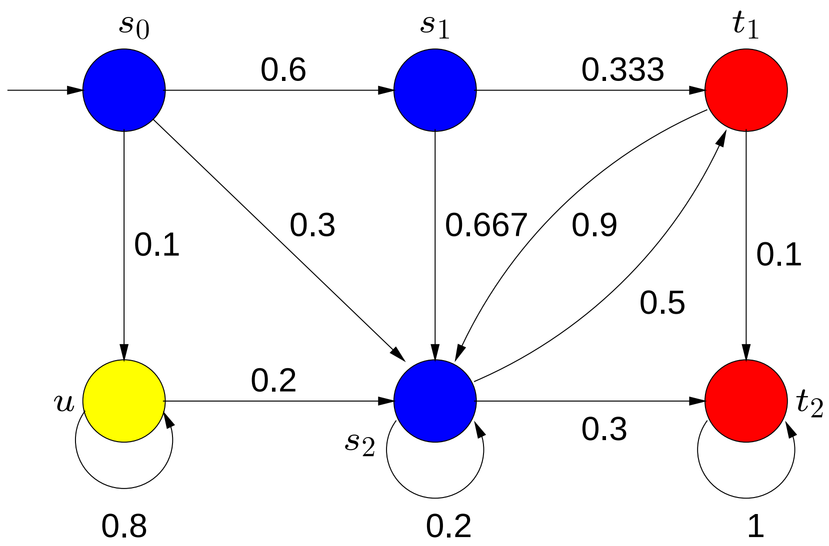
Counterexamples for $s_0 \not\models \mathbb{P}_{\leqslant \frac{1}{2}}(a \, \mathsf{U} \, b)$



evidences	prob.
$\sigma_1 = s_0\, s_1\, t_1$	0.2
$\sigma_2 = s_0\, s_1\, s_2\, t_1$	0.2
$\sigma_3 = s_0\, s_2\, t_1$	0.15
$\sigma_4 = s_0\, s_1\, s_2\, t_2$	0.12
$\sigma_5 = s_0\, s_2\, t_2$	0.09

counterexample	card.	prob.
$\{ \sigma_1, \ldots, \sigma_5 \}$	5	0.76
$\{ \sigma_1$ or $\sigma_2, \ldots, \sigma_5 \}$	4	0.56
$\{ \sigma_1, \sigma_2, \sigma_4 \}$	3	0.52
$\{ \sigma_1, \sigma_2, \sigma_3 \}$	3	0.55

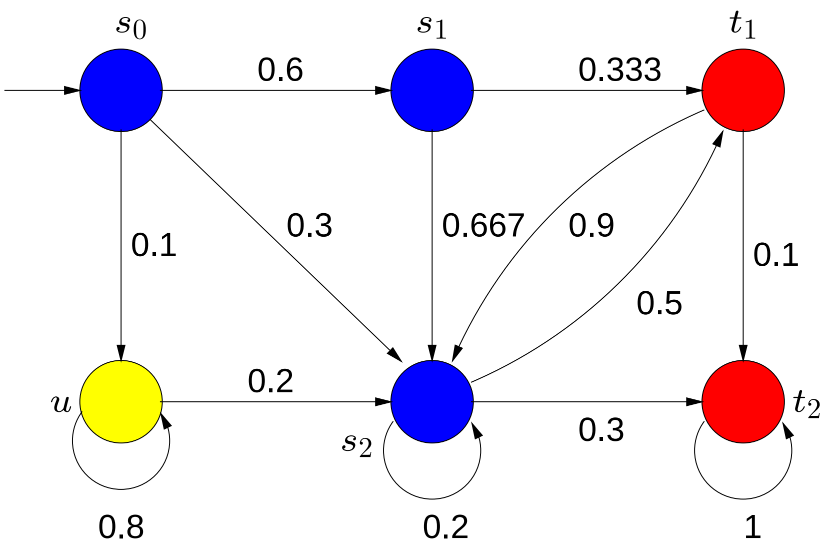
Counterexamples for $s_0 \not\models \mathbb{P}_{\leqslant \frac{1}{2}}(a \,\mathsf{U}\, b)$



evidences	prob.
$\sigma_1 = s_0\, s_1\, t_1$	0.2
$\sigma_2 = s_0\, s_1\, s_2\, t_1$	0.2
$\sigma_3 = s_0\, s_2\, t_1$	0.15
$\sigma_4 = s_0\, s_1\, s_2\, t_2$	0.12
$\sigma_5 = s_0\, s_2\, t_2$	0.09

counterexample	card.	prob.
$\{ \sigma_1, \ldots, \sigma_5 \}$	5	0.76
$\{ \sigma_1$ or $\sigma_2, \ldots, \sigma_5 \}$	4	0.56
minimal $\longrightarrow \{ \sigma_1, \sigma_2, \sigma_4 \}$	3	0.52
minimal $\longrightarrow \{ \sigma_1, \sigma_2, \sigma_3 \}$	3	0.55

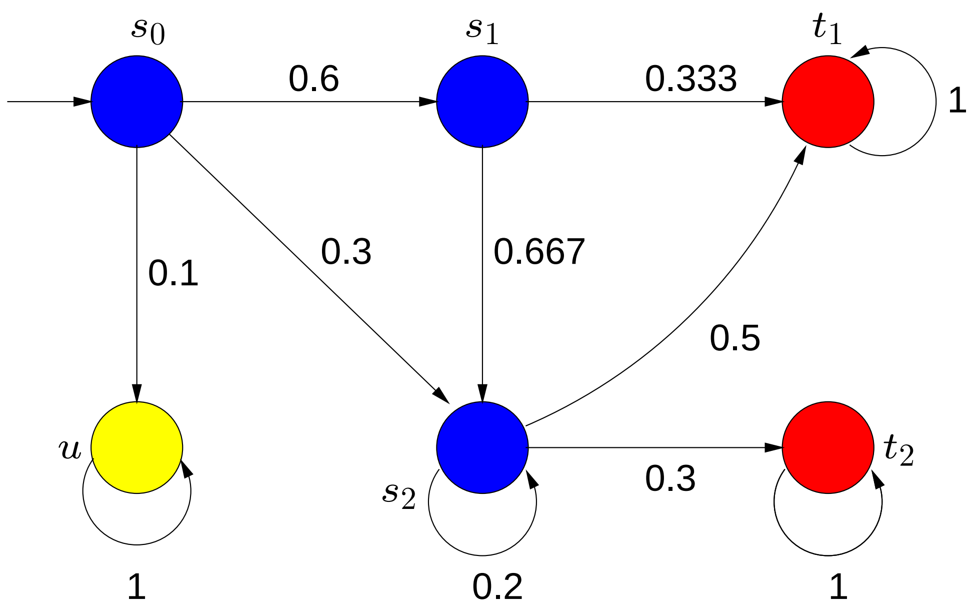
Counterexamples for $s_0 \not\models \mathbb{P}_{\leqslant \frac{1}{2}}(a \, \mathsf{U} \, b)$



evidences	prob.
$\sigma_1 = s_0 \, s_1 \, t_1$	0.2
$\sigma_2 = s_0 \, s_1 \, s_2 \, t_1$	0.2
$\sigma_3 = s_0 \, s_2 \, t_1$	0.15
$\sigma_4 = s_0 \, s_1 \, s_2 \, t_2$	0.12
$\sigma_5 = s_0 \, s_2 \, t_2$	0.09

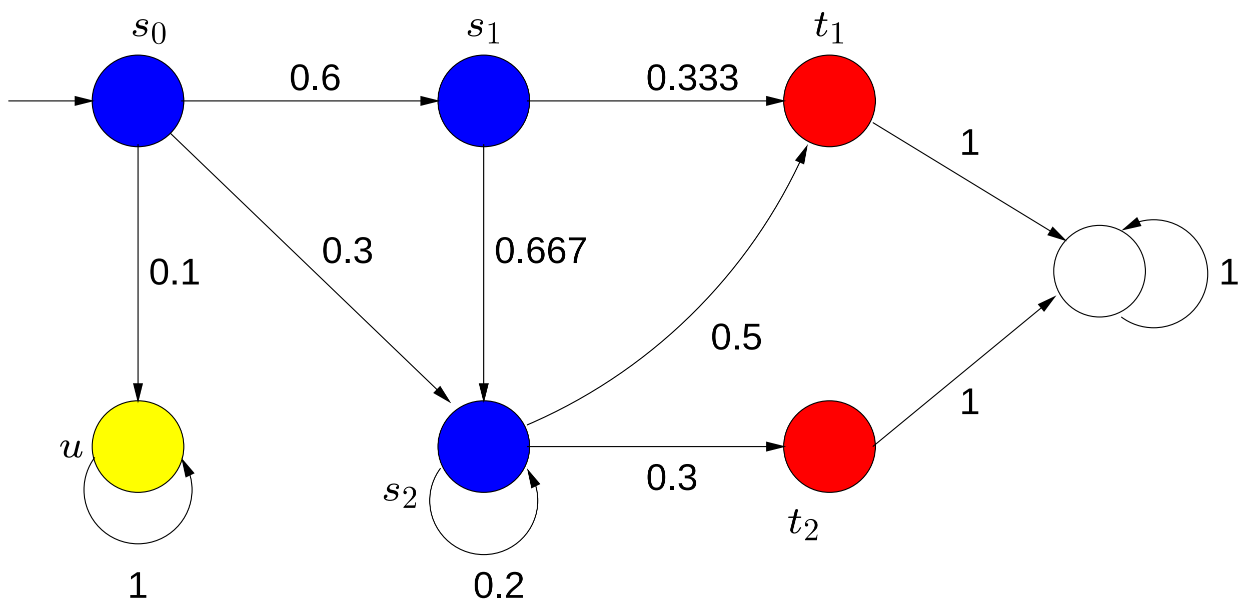
counterexample	card.	prob.
$\{ \sigma_1, \ldots, \sigma_5 \}$	5	0.76
$\{ \sigma_1$ or $\sigma_2, \ldots, \sigma_5 \}$	4	0.56
$\{ \sigma_1, \sigma_2, \sigma_4 \}$	3	0.52
smallest $\longrightarrow$ $\{ \sigma_1, \sigma_2, \sigma_3 \}$	3	0.55

Obtaining smallest counterexamples



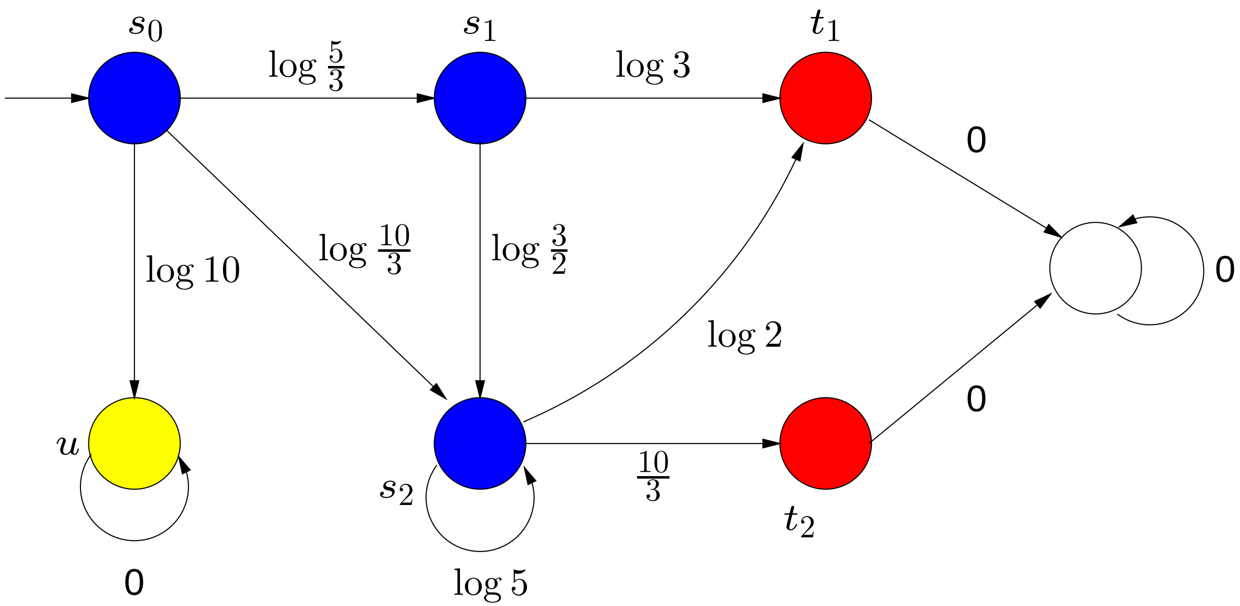
Step 1: make all Ψ-states and all $\neg\Phi \wedge \neg\Psi$-states absorbing

Adapting a bit more



Step 2: insert a sink state and redirect all outgoing edges of Ψ-states to it

A weighted digraph



Step 3: turn it into a weighted digraph with $w(s, s') = \log \left(\dfrac{1}{\mathbf{P}(s, s')} \right)$

A simple derivation

For finite path $\sigma = s_0\, s_1\, s_2\, \ldots\, s_n$:

$$
\begin{aligned}
w(\sigma) &= w(s_0, s_1) + w(s_1, s_2) + \ldots + w(s_{n-1}, s_n) \\
&= \log \frac{1}{\mathbf{P}(s_0,s_1)} + \log \frac{1}{\mathbf{P}(s_1,s_2)} + \ldots + \log \frac{1}{\mathbf{P}(s_{n-1},s_n)} \\
&= \log \frac{1}{\mathbf{P}(s_0,s_1)\cdot\mathbf{P}(s_1,s_2)\cdot\ldots\cdot\mathbf{P}(s_{n-1},s_n)} \\
&= \log \frac{1}{\Pr(\sigma)}
\end{aligned}
$$

$$\underbrace{\Pr(\widehat{\sigma}) \geqslant \Pr(\sigma)}_{\text{in DTMC } \mathcal{D}} \quad \text{if and only if} \quad \underbrace{w(\widehat{\sigma}) \leqslant w(\sigma)}_{\text{in digraph } G(\mathcal{D})}$$

What does this mean?

- Finding a strongest evidence is a shortest path (SP) problem
  - apply standard SP algorithms, or Viterbi's algorithm $\Rightarrow$ linear time complexity

What does this mean?

- Finding a strongest evidence is a shortest path (SP) problem
  - apply standard SP algorithms, or Viterbi's algorithm $\Rightarrow$ linear time complexity
- Finding a shortest counterex is a k-shortest path (KSP) problem
  - dynamically determine k: generate C incrementally and halt when $\Pr(C) > p$

What does this mean?

- Finding a strongest evidence is a shortest path (SP) problem
  - apply standard SP algorithms, or Viterbi's algorithm $\Rightarrow$ linear time complexity
- Finding a shortest counterex is a k-shortest path (KSP) problem
  - dynamically determine k: generate C incrementally and halt when $\Pr(C) > p$
- This also applies to $\mathbb{P}_{\geqslant p}(\varphi)$ properties, as

$$\mathbb{P}_{\geqslant p}(\Phi \, \mathsf{U} \, \Psi) \equiv \mathbb{P}_{\leqslant 1-p}(\underbrace{(\Phi \wedge \neg \Psi)}_{\Phi^*} \, \mathsf{W} \, \underbrace{(\neg \Phi \wedge \neg \Psi)}_{\Psi^*})$$

$$\equiv \mathbb{P}_{\leqslant 1-p}(\Phi^* \, \mathsf{U} \, (\Psi^* \vee \mathit{at}_{\mathsf{bscc}(\Phi^*)}))$$

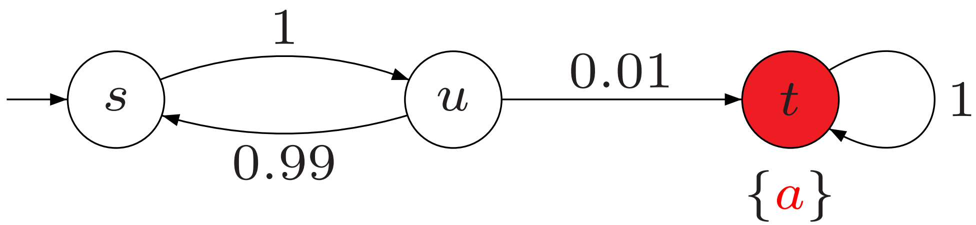
Time complexity

counterexample problem	shortest path problem	algorithm	time complexity
unbounded SE	SP	Dijkstra	$\mathcal{O}(M + N \cdot \log N)$
bounded h SE	HSP	Bellman-Ford / Viterbi	$\mathcal{O}(h \cdot M)$
unbounded SC	KSP	Eppstein	$\mathcal{O}(M + N \cdot \log N + k)$
bounded h SC	HKSP	adapted REA	$\mathcal{O}\left(h \cdot M + h \cdot k \cdot \log N\right)$

N = $|S|$, $M = \#$ transitions, h = hop count, k = $\#$ shortest paths

including costs yields an instance of the NP-complete RSP problem

On the size of counterexamples



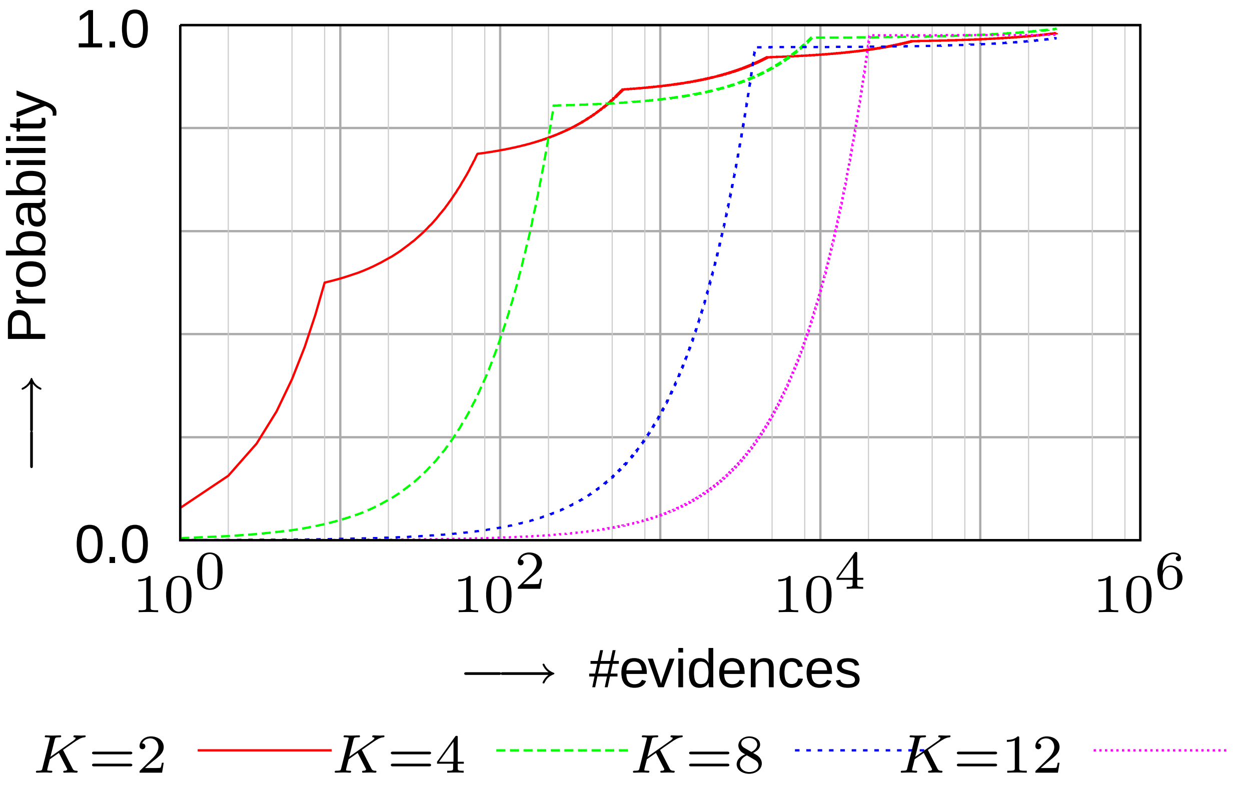
A smallest counterexample for $s \not\models \mathbb{P}_{\leqslant 0.9999}(\Diamond\, a)$ contains paths

$$s\,u\,t,\; s\,u\,s\,u\,t,\; s\,u\,s\,u\,s\,u\,t,\; \ldots\ldots,\; \underbrace{s\,u}_{k \text{ times}}\;t$$

where k is the smallest integer such that $1 - 0.99^{k-1} > 0.9999$

The smallest counterexample has $k = 689$ evidences

Synchronous leader election $\mathbb{P}_{\leqslant 0.99}(\Diamond\, \textit{leader})$

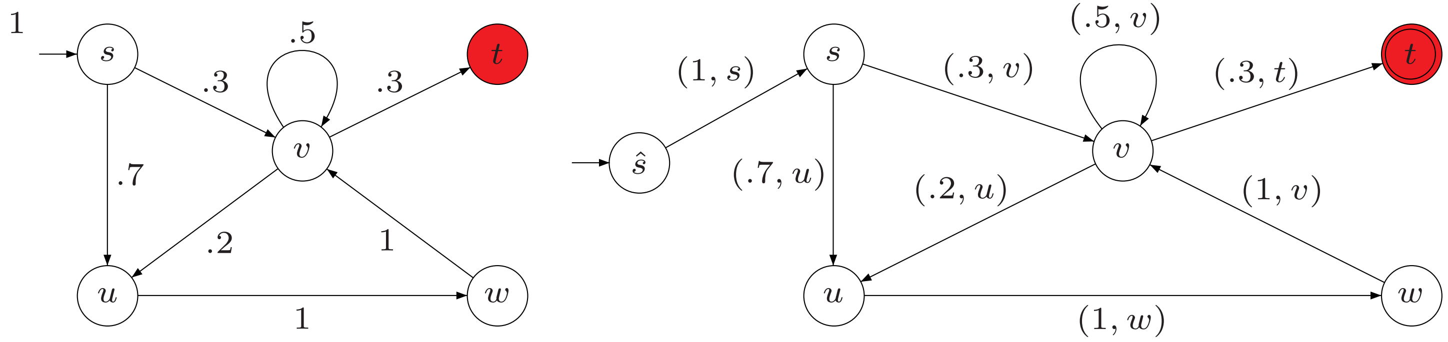


size of counterexample is double exponential in problem size (see paper)

Use regular expressions!

- Size of counterexamples is mainly influenced by loops
  - each loop-traversal yields another path in counterexample
- Idea: represent sets of “similar” finite paths by a *regular expression*
- How?
  - DTMC (rooted at s) $\longrightarrow$ DFA
  - DFA $\longrightarrow$ most probable paths $\longrightarrow$ regular expression r
- Such that:
  - probability of regular expression r exceeds p (= r is a counterexample)
  - r is “minimal”: deletion of some “branch” of r yields no counterexample

From DTMCs to DFAs



alphabet Σ consist of symbols of the form (p, s)

From DTMCs to DFA

For DTMC $\mathcal{D} = (S, \mathbf{P}, L)$, state s, and property $\mathbb{P}(\Diamond^{\leqslant h} t)$, DFA $\mathcal{A}_{\mathcal{D}} = (S', \Sigma, \tilde{s}, \delta, t)$

	DTMC	DFA
state space	S	$S \cup \{\tilde{s}\}$
initial state	s	$\tilde{s} \notin S$
goal/accepting state	t	t
alphabet	–	$\Sigma \subset [0,1] \times S$
transitions	$s_1 \xrightarrow{p} s_2$	$s_1 \xrightarrow{(p, s_2)} s_2$
	–	$\tilde{s} \xrightarrow{(1,s)} s$

Regular expressions [Daws'04]

The set of regular expressions $\mathcal{R}(\Sigma)$:

$$
\begin{array}{rcll}
r, r' & ::= & \varepsilon & \text{empty} \\
& | & (p, s) & \text{letter} \\
& | & r|r' & \text{choice} \\
& | & r.r' & \text{catenation} \\
& | & r^* & \text{repetition}
\end{array}
$$

Regular expressions [Daws'04]

The set of regular expressions $\mathcal{R}(\Sigma)$:

$$
\begin{array}{rcll}
r, r' & ::= & \varepsilon & \text{empty} \\
& | & (p, s) & \text{letter} \\
& | & r|r' & \text{choice} \\
& | & r.r' & \text{catenation} \\
& | & r^* & \text{repetition}
\end{array}
$$

Evaluation $val : \mathcal{R}(\Sigma) \to [0,1]$:

$$
\begin{array}{rcl}
val(\varepsilon) & = & 1 \\
val((p, s)) & = & p \\
val(r|r') & = & val(r) + val(r') \\
val(r.r') & = & val(r) \cdot val(r') \\
val(r^*) & = & \begin{cases} 1 & \text{if } val(r) = 1 \\ \frac{1}{1 - val(r)} & \text{otherwise} \end{cases}
\end{array}
$$

Regular expressions [Daws'04]

The set of regular expressions $\mathcal{R}(\Sigma)$:

$$
\begin{array}{rcll}
r, r' & ::= & \varepsilon & \text{empty} \\
& \mid & (p, s) & \text{letter} \\
& \mid & r | r' & \text{choice} \\
& \mid & r.r' & \text{catenation} \\
& \mid & r^* & \text{repetition}
\end{array}
$$

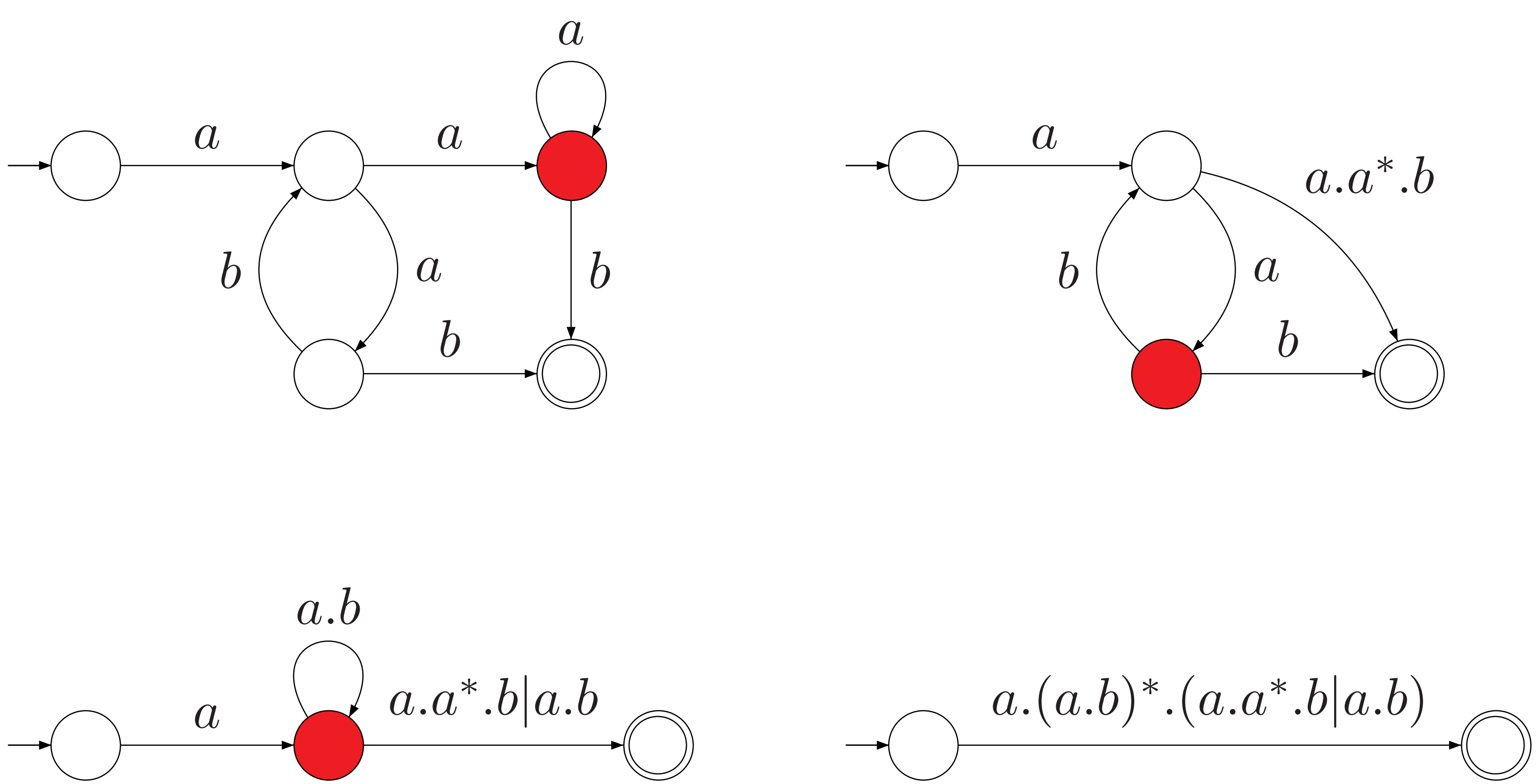
Evaluation $val : \mathcal{R}(\Sigma) \rightarrow [0, 1]$:

$$
\begin{array}{rcl}
val(\varepsilon) & = & 1 \\
val((p, s)) & = & p \\
val(r|r') & = & val(r) + val(r') \\
val(r.r') & = & val(r) \cdot val(r') \\
val(r^*) & = & \begin{cases} 1 & \text{if } val(r) = 1 \\ \frac{1}{1 - val(r)} & \text{otherwise} \end{cases}
\end{array}
$$

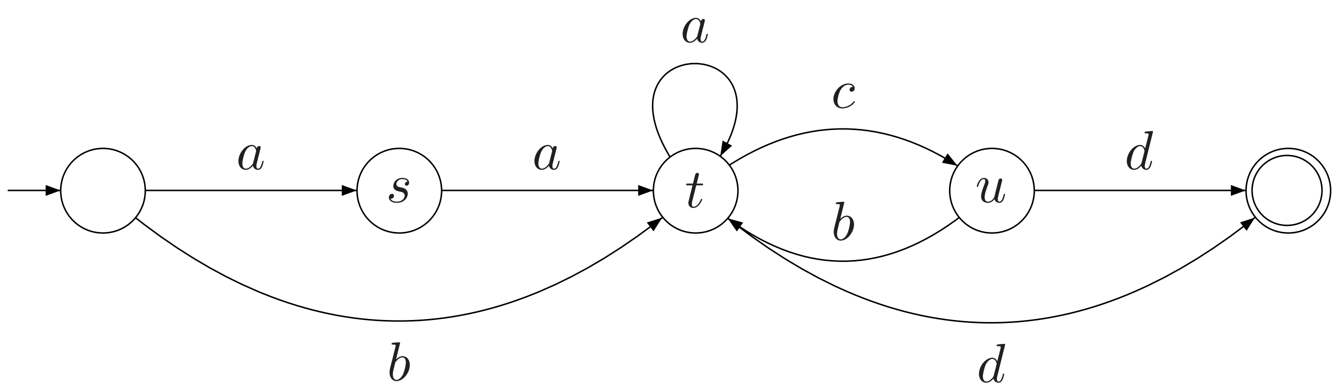
For regular expression r of DFA $\mathcal{A}_{\mathcal{D}}$ with accept state t:

$$val(r) \; = \; \Pr{}^{\mathcal{D}} \{ \sigma \in \mathit{Paths}(s) \mid \sigma \models \Diamond\, t \}$$

State elimination [Brzozowski & McCluskey jr., 1962]



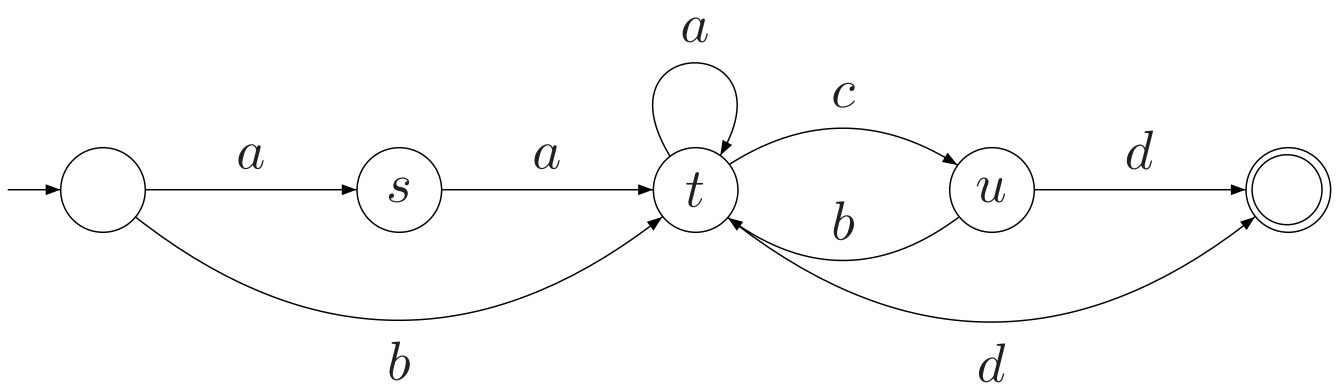
Ordering matters



Ordering $s < u < t$ yields $(aa|b)(a|cb)^*(cd|d)$

Ordering $s < t < u$ yields $(aa|b)a^*c(ba^*c)^*(ba^*d|d)|(aa|b)a^*d$

Ordering matters



Finding the optimal removal ordering takes time $\mathcal{O}(N!)$ where $|S| = N$

Heuristic [Han & Wood'07]

"eliminate all non-bridge states before bridge states"

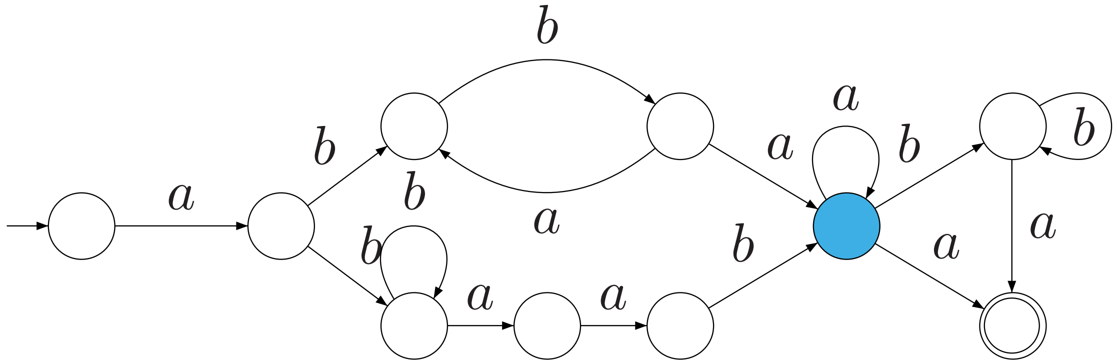
1. Find all *bridge* states q_1 through q_{n-1}
   - the path of every word $w \in \mathcal{L}(\mathcal{A})$ goes through q_i
   - once this path visits q_i it will not visit states visited prior to q_i
2. Perform *vertical chopping*
   - $\mathcal{A} = \mathcal{A}_1 \cdot \mathcal{A}_2 \cdot \ldots \cdot \mathcal{A}_n$ where $\mathcal{A}_i$ is "connected" to $\mathcal{A}_i$ via bridge q_i
3. For each $\mathcal{A}_i$ perform *horizontal chopping*
   - $\mathcal{A}_i = \mathcal{A}_{i,1} | \mathcal{A}_{i,2} | \ldots | \mathcal{A}_{i,k}$
4. For each automaton $\mathcal{A}_i, j$ goto step 1.

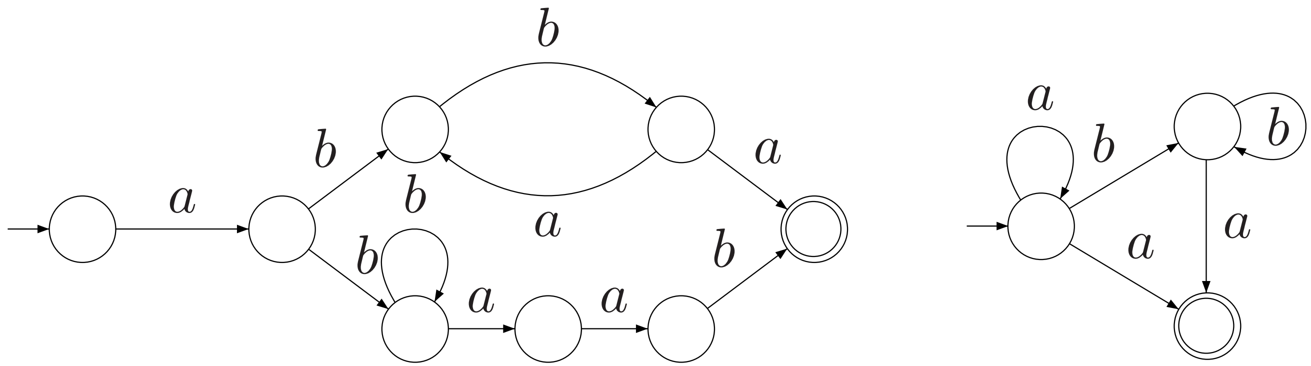
Time complexity

"eliminate all non-bridge states before bridge states"

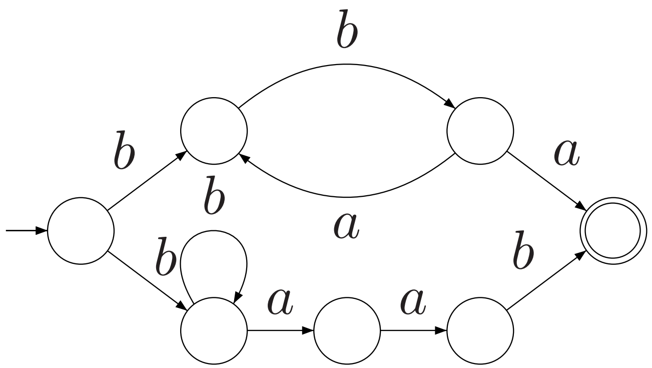
1. Find all *bridge* states q_1 through q_{n-1} — in linear time
   - the path of every word $w \in \mathcal{L}(\mathcal{A})$ goes through q_i
   - once this path visits q_i it will not visit states visited prior to q_i
2. Perform *vertical chopping* — in linear time
   - $\mathcal{A} = \mathcal{A}_1 \cdot \mathcal{A}_2 \cdot \ldots \cdot \mathcal{A}_n$ where $\mathcal{A}_i$ is "connected" to $\mathcal{A}_i$ via bridge q_i
3. For each $\mathcal{A}_i$ perform *horizontal chopping* — in linear time
   - $\mathcal{A}_i = \mathcal{A}_{i,1} \mid \mathcal{A}_{i,2} \mid \ldots \mid \mathcal{A}_{i,k}$
4. For each automaton $\mathcal{A}_{i,j}$ goto step 1.

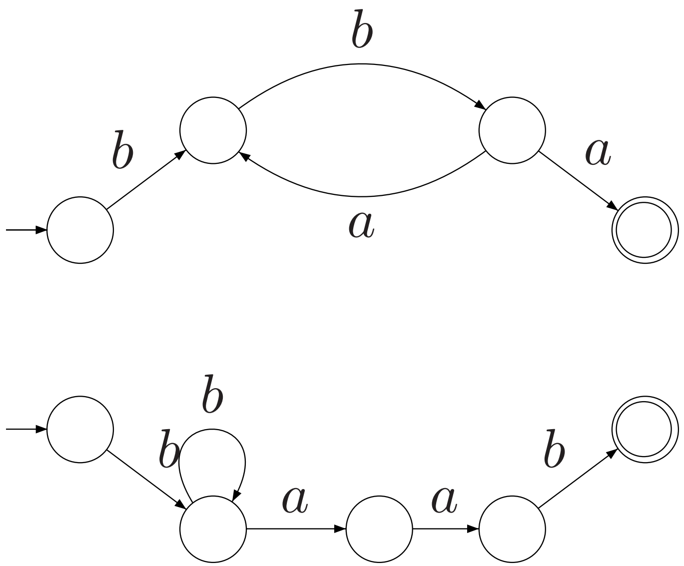
Vertical chopping





Horizontal chopping





Maximal union subexpressions

r_1 is a *maximal union subexpression* (MUS) of regular expression r if:

$$r \;=\; r_1 \mid r_2 \qquad \text{modulo the congruence } (\mathbf{R_1})\text{-}(\mathbf{R_3})$$

where for some $r_2 \in \mathcal{R}(\Sigma)$:

$$\begin{array}{lrcl} (\mathbf{R_1}) & r & \equiv & r \mid \varepsilon \\ (\mathbf{R_2}) & r_1 \mid r_2 & \equiv & r_2 \mid r_1 \\ (\mathbf{R_3}) & r_1 \mid (r_2 \mid r_3) & \equiv & (r_1 \mid r_2) \mid r_3 \end{array}$$

a MUS can be regarded as a main path from the initial state to a accept state

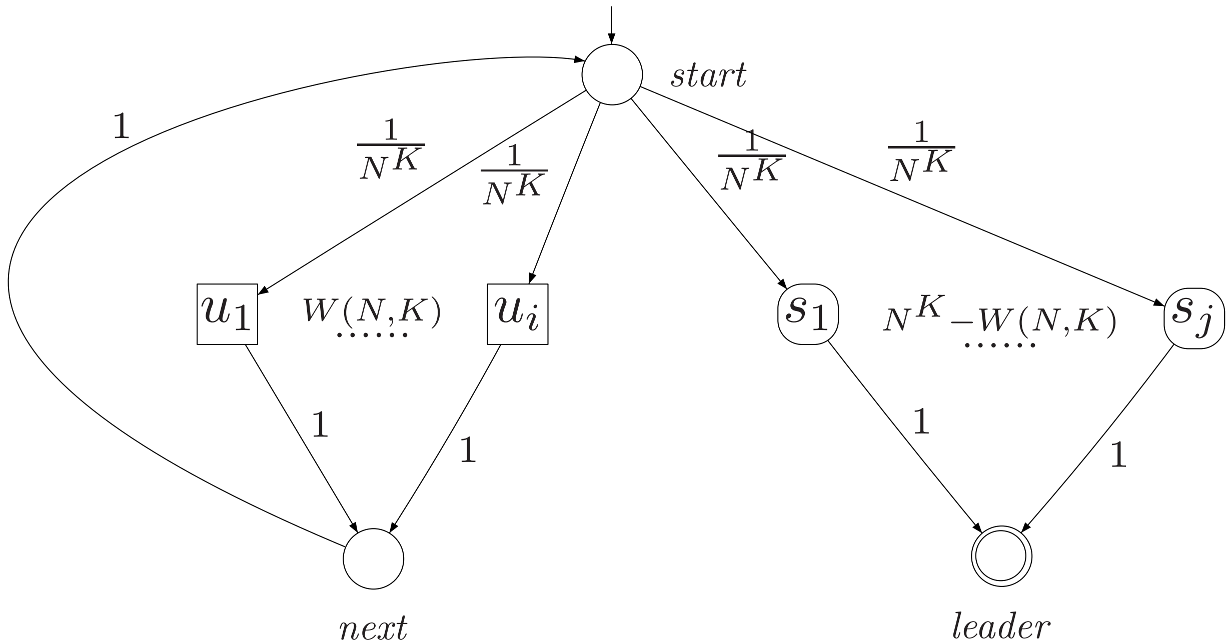
Algorithm for regular expressions

Require: DFA $\mathcal{A}_{\mathcal{D}} = (S, \Sigma, s, \delta, \{t\})$, and $p \in [0,1]$
Ensure: regular expression $r \in \mathcal{R}(\Sigma)$ with $val(r) > p$

$\mathcal{A} := \mathcal{A}_{\mathcal{D}}$, $pr := 0$; priority queue $pq := \varnothing$; $k := 1$;
while $pr \leqslant p$ **do**
  $\sigma :=$ the strongest evidence in $\mathcal{A}$;
  forall $s' \in \sigma \setminus \{s, \hat{s}, t\}$ **do** pq.enqueue(s'); **end**;
  while $pq \neq \varnothing$ **do**
    $\mathcal{A}$:=eliminate(pq.dequeue()); $r_k :=$ the created MUS;
    $pr := pr + val(r_k)$; $\mathcal{A}$:=eliminate(r_k);
    if ($pr > p$) **then break** **else** $k := k + 1$;
  endwhile;
endwhile;
return $r_1 \mid \ldots \mid r_k$.

this approach works for strict and non-strict bounds

Leader election revisited



Regular expression for the counterexample:

$$r(N, K) = start.\ [(u_1 | \cdots | u_i)\ .next.\ start]^*.\ (s_1 | \cdots | s_j).\ leader$$

Model reduction

The size of a counterexample is determined by

- traversing the same loop for different times

  $\Longrightarrow$ using Kleene stars in regular expressions

- large number of states

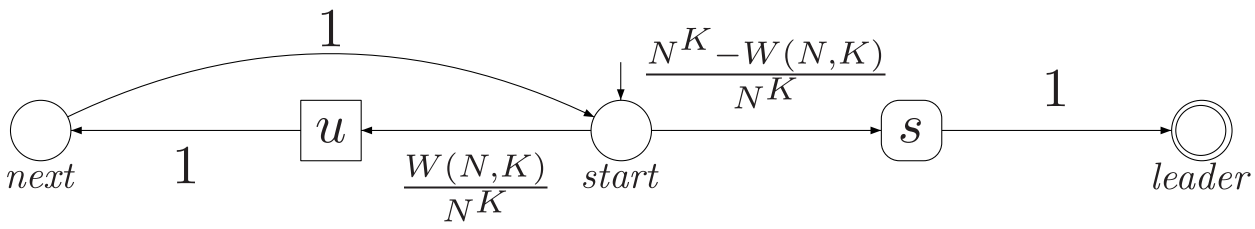
  $\Longrightarrow$ model reduction

  1. bisimulation minimization
  2. SCC minimization

Model reduction is done prior to counterexample generation

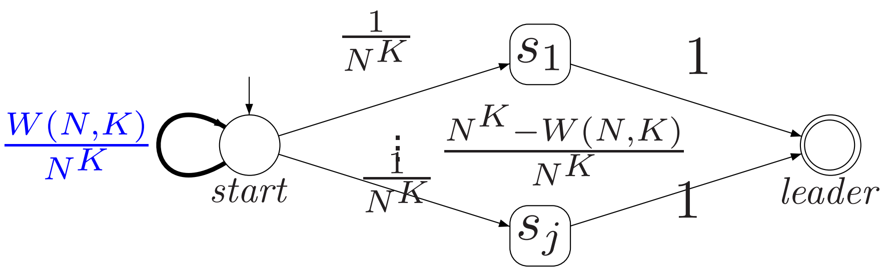
Leader election re-revisited

Bisimulation quotient:



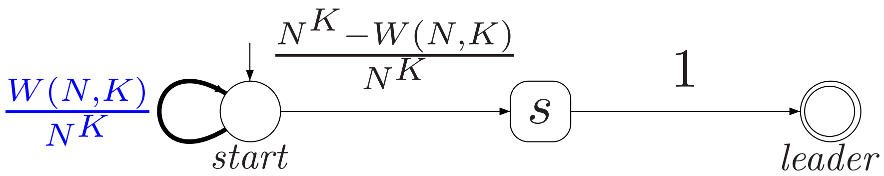
$$r_{\sim}(N,K) = start.\,(u.next.start)^{*}.s.\;leader$$

After aggregating SCCs:



$$r^{scc}(N,K) = start.start^{*}.(s_1|\cdots|s_j).leader$$

SCC aggregation of bisimulation quotient:



$$r^{scc}_{\sim}(N,K) = start.start^{*}.s.leader$$

Counterexamples are *en vogue*

- Heuristic search algorithms for CTMCs (Aljazzar *et al.* FORMATS 2005, 2006)
- Counterexamples for CTMCs (Han & Katoen ATVA 2007)
- Counterexamples for conditional PCTL (Andres & van Rossum TACAS 2008)
- Proof refutations for probabilistic programs (McIver *et al.* FM 2008)
- Counterexample-guided abstraction refinement (Hermanns *et al.* CAV 2008) (Chadha & Viswamanathan TR 2008)
- Counterexamples for MDPs (Andres *et al.*, HVC 2008, Aljazzar & Leue TR 2007)
- Bounded model checking for DTMC counterexamples (Becker *et al.* TR 2008)

Epilogue

- What is a PCTL (or quantitative LTL) counterexample?
  - a set of paths with sufficient probability mass
- How to determine smallest counterexamples?
  - exploit k-shortest path algorithms
- How about the size of counterexamples?
  - well, they may be excessively large and incomprehensible
- Can we do better?
  - yes, represent counterexamples by regular expressions!
- How to obtain (short) regular expressions?
  - use automata theory and some heuristics

谢谢大家！