Combining Decomposition and Reduction for State Space Analysis of a Self-Stabilizing System

Nils Müllner

nils.muellner@informatik.uni-oldenburg.de
Abteilung Systemsoftware und verteilte Systeme
Department für Informatik
Carl von Ossietzky Universität Oldenburg

29. March 2012

outline

self-stabilizing broadcast

$\underline{\text{const}}\ id := 0,$
$\underline{\text{var}}\ reg,$
$\text{repeat}\{$
$\quad reg := \mathbf{0}\}$

Figure: Root Process

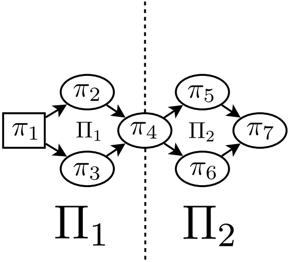
$\underline{\text{const}}\ neighbors := \{\pi_i, \ldots\},$
$\underline{\text{const}}\ id := \min\{id(\pi_i), \ldots\} + 1,$
$\underline{\text{var}}\ reg,$
$\underline{\text{var}}\ set := reg_i, \pi(reg_i) \in$
$\quad neighbors | \forall i : id(\pi_i) = id - 1\}$
$\text{repeat}\{$
$\neg((\exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{2})$
$xor(\exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{0}))$
$\quad \rightarrow reg := \mathbf{1}$
$\Box \exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{0}$
$\quad \rightarrow reg := \mathbf{0}$
$\Box \exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{2}$
$\quad \rightarrow reg := \mathbf{2}\}$

Figure: Broadcast Sub-Algorithm for Non-Root Processes

self-stabilizing broadcast

$\underline{\text{const}}\ id := 0,$
$\underline{\text{var}}\ reg,$
$\text{repeat}\{$
$\quad reg := \mathbf{0}\}$

Figure: Root Process



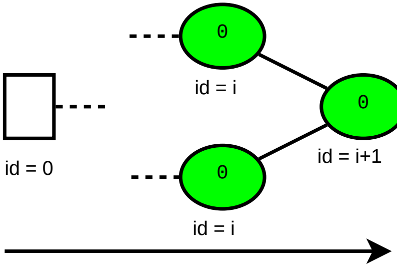
$\underline{\text{const}}\ neighbors := \{\pi_i, \ldots\},$
$\underline{\text{const}}\ id := \min\{id(\pi_i), \ldots\} + 1,$
$\underline{\text{var}}\ reg,$
$\underline{\text{var}}\ set := reg_i, \pi(reg_i) \in$
$\quad neighbors | \forall i : id(\pi_i) = id - 1\}$
$\text{repeat}\{$
$\neg((\exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{2})$
$xor(\exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{0}))$
$\quad \rightarrow reg := \mathbf{1}$
$\square \exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{0}$
$\quad \rightarrow reg := \mathbf{0}$
$\square \exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{2}$
$\quad \rightarrow reg := \mathbf{2}\}$

Figure: Broadcast Sub-Algorithm for Non-Root Processes

self-stabilizing broadcast

$\underline{\text{const}}\ id := 0,$
$\underline{\text{var}}\ reg,$
$\text{repeat}\{$
$\quad reg := \mathbf{0}\}$

Figure: Root Process



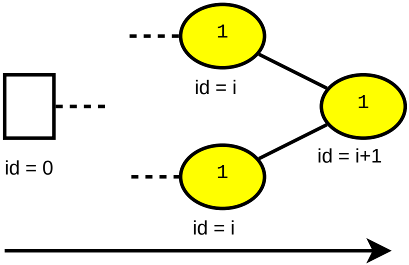
$\underline{\text{const}}\ neighbors := \{\pi_i, \ldots\},$
$\underline{\text{const}}\ id := \min\{id(\pi_i), \ldots\} + 1,$
$\underline{\text{var}}\ reg,$
$\underline{\text{var}}\ set := reg_i, \pi(reg_i) \in$
$\quad neighbors | \forall i : id(\pi_i) = id - 1\}$
$\text{repeat}\{$
$\neg((\exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{2})$
$xor(\exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{0}))$
$\quad \rightarrow reg := \mathbf{1}$
$\Box \exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{0}$
$\quad \rightarrow reg := \mathbf{0}$
$\Box \exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{2}$
$\quad \rightarrow reg := \mathbf{2}\}$

Figure: Broadcast Sub-Algorithm for Non-Root Processes

self-stabilizing broadcast

$\underline{\text{const}}$ $id := 0,$
$\underline{\text{var}}$ $reg,$
repeat$\{$
$reg := \mathbf{0}\}$

Figure: Root Process



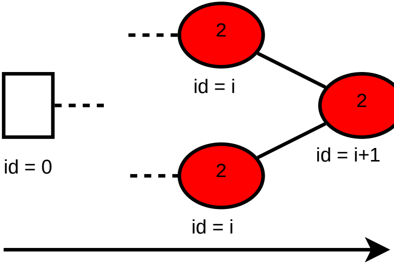
$\underline{\text{const}}$ $neighbors := \{\pi_i, \ldots\},$
$\underline{\text{const}}$ $id := \min\{id(\pi_i), \ldots\} + 1,$
$\underline{\text{var}}$ $reg,$
$\underline{\text{var}}$ $set := reg_i, \pi(reg_i) \in$
$neighbors | \forall i : id(\pi_i) = id - 1\}$
repeat$\{$
$\neg((\exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{2})$
$xor(\exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{0}))$
$\rightarrow reg := \mathbf{1}$
$\Box \exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{0}$
$\rightarrow reg := \mathbf{0}$
$\Box \exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{2}$
$\rightarrow reg := \mathbf{2}\}$

Figure: Broadcast Sub-Algorithm for Non-Root Processes

self-stabilizing broadcast

$\underline{\text{const}}\ id := 0,$
$\underline{\text{var}}\ reg,$
$\text{repeat}\{$
$\quad reg := \mathbf{0}\}$

Figure: Root Process



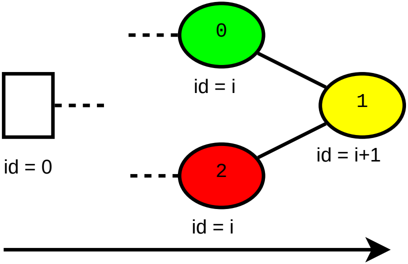
$\underline{\text{const}}\ neighbors := \{\pi_i, \ldots\},$
$\underline{\text{const}}\ id := \min\{id(\pi_i), \ldots\} + 1,$
$\underline{\text{var}}\ reg,$
$\underline{\text{var}}\ set := reg_i, \pi(reg_i) \in$
$\quad neighbors | \forall i : id(\pi_i) = id - 1\}$
$\text{repeat}\{$
$\neg((\exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{2})$
$xor(\exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{0}))$
$\quad \rightarrow reg := \mathbf{1}$
$\Box \exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{0}$
$\quad \rightarrow reg := \mathbf{0}$
$\Box \exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{2}$
$\quad \rightarrow reg := \mathbf{2}\}$

Figure: Broadcast Sub-Algorithm for Non-Root Processes

self-stabilizing broadcast

$\underline{\text{const}}\ id := 0,$
$\underline{\text{var}}\ reg,$
$\text{repeat}\{$
$\quad reg := \mathbf{0}\}$

Figure: Root Process



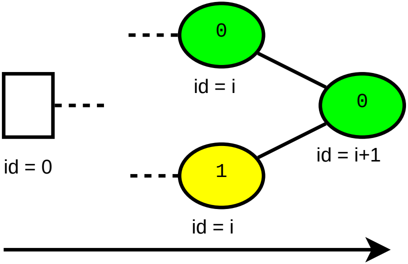
$\underline{\text{const}}\ neighbors := \{\pi_i, \ldots\},$
$\underline{\text{const}}\ id := \min\{id(\pi_i), \ldots\} + 1,$
$\underline{\text{var}}\ reg,$
$\underline{\text{var}}\ set := reg_i, \pi(reg_i) \in$
$\quad neighbors | \forall i : id(\pi_i) = id - 1\}$
$\text{repeat}\{$
$\neg((\exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{2})$
$xor(\exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{0}))$
$\quad \rightarrow reg := \mathbf{1}$
$\square \exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{0}$
$\quad \rightarrow reg := \mathbf{0}$
$\square \exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{2}$
$\quad \rightarrow reg := \mathbf{2}\}$

Figure: Broadcast Sub-Algorithm for Non-Root Processes

self-stabilizing broadcast

$\underline{\text{const}}$ $id := 0$,
$\underline{\text{var}}$ reg,
repeat$\{$
$reg := \mathbf{0}\}$

Figure: Root Process



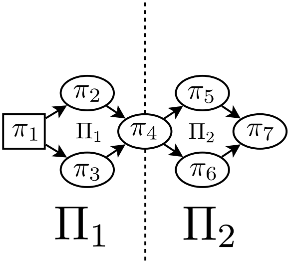
$\underline{\text{const}}$ $neighbors := \{\pi_i, \ldots\}$,
$\underline{\text{const}}$ $id := \min\{id(\pi_i), \ldots\} + 1$,
$\underline{\text{var}}$ reg,
$\underline{\text{var}}$ $set := reg_i, \pi(reg_i) \in neighbors | \forall i : id(\pi_i) = id - 1\}$
repeat$\{$
$\neg((\exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{2})$
$xor(\exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{0}))$
$\rightarrow reg := \mathbf{1}$
$\square \exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{0}$
$\rightarrow reg := \mathbf{0}$
$\square \exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{2}$
$\rightarrow reg := \mathbf{2}\}$

Figure: Broadcast Sub-Algorithm for Non-Root Processes

self-stabilizing broadcast

$\underline{\text{const}}\ id := 0,$
$\underline{\text{var}}\ reg,$
$\text{repeat}\{$
$\quad reg := \mathbf{0}\}$

Figure: Root Process



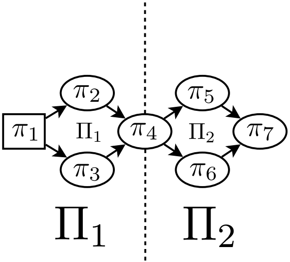
$\underline{\text{const}}\ neighbors := \{\pi_i, \ldots\},$
$\underline{\text{const}}\ id := \min\{id(\pi_i), \ldots\} + 1,$
$\underline{\text{var}}\ reg,$
$\underline{\text{var}}\ set := reg_i, \pi(reg_i) \in$
$\quad neighbors | \forall i : id(\pi_i) = id - 1\}$
$\text{repeat}\{$
$\neg((\exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{2})$
$xor(\exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{0}))$
$\quad \rightarrow reg := \mathbf{1}$
$\square \exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{0}$
$\quad \rightarrow reg := \mathbf{0}$
$\square \exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{2}$
$\quad \rightarrow reg := \mathbf{2}\}$

Figure: Broadcast Sub-Algorithm for Non-Root Processes

self-stabilizing broadcast

$\underline{\text{const}}$ $id := 0$,
$\underline{\text{var}}$ reg,
repeat$\{$
$\quad reg := \mathbf{0}\}$

Figure: Root Process



$s_t \models \mathcal{P} : reg_1 = \mathbf{0} \wedge \ldots \wedge reg_7 = \mathbf{0}$

$\underline{\text{const}}$ $neighbors := \{\pi_i, \ldots\}$,
$\underline{\text{const}}$ $id := \min\{id(\pi_i), \ldots\} + 1$,
$\underline{\text{var}}$ reg,
$\underline{\text{var}}$ $set := reg_i, \pi(reg_i) \in$
$\quad neighbors | \forall i : id(\pi_i) = id - 1\}$
repeat$\{$
$\neg((\exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{2})$
$xor(\exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{0}))$
$\quad \rightarrow reg := \mathbf{1}$
$\Box \exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{0}$
$\quad \rightarrow reg := \mathbf{0}$
$\Box \exists reg_i : \pi(reg_i) \in set \wedge reg_i = \mathbf{2}$
$\quad \rightarrow reg := \mathbf{2}\}$

Figure: Broadcast Sub-Algorithm for Non-Root Processes

Limiting Window Availability

probability, that, from the limit, $s \models \mathcal{P}$ for one step within window

Limiting Window Availability

Definition 1 (Limiting Window Availability (LWA))

Assume that at time $t = 0$, an initial distribution holds that corresponds to the stationary distribution of a system. Then, Limiting Window Availability of window size w (of this system), denoted by LWA_w, $w \geq 0$, is the probability that the system has at least once reached a state satisfying $\mathcal{P}$ within the following w time steps:

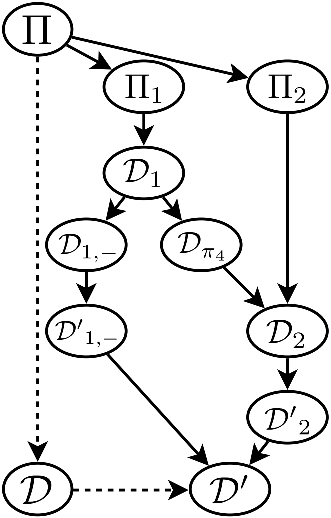
$$LWA_w = prob\{\exists k, 0 \leq k \leq w : s_k \models \mathcal{P}\}$$

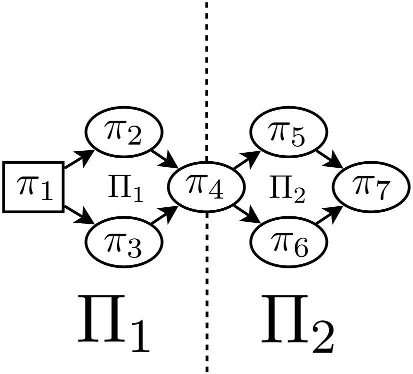
w is called window size.

Trivia

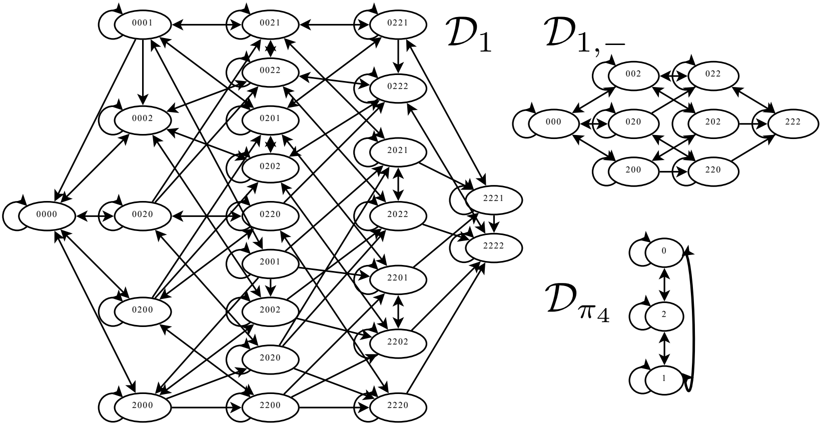
- has 648 states: $2 * 2 * 2 * 3 * 3 * 3 * 3$, $\langle \mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0} \rangle, \ldots, \langle \mathbf{2},\mathbf{2},\mathbf{2},\mathbf{2},\mathbf{2},\mathbf{2},\mathbf{2} \rangle$
- probabilistic scheduler
- serial execution semantics (to exclude hazards)
- transient faults $q = 0.05$
- processes [π_2 with π_3], and [π_5 with π_6], are strongly probabilistic bisimilar
- lumping allows for reduction to 324 states

Plan of Action





Plan of Action



Plan of Action

- handling at most 81 state at a time
  until recomposition
- self-stabilization transforms transition model
  into a DAG
- exploitable symmetries in heterarchical systems

the Markov chain $\mathcal{D}$ to compute the *LWA*

$\downarrow$from/to$\rightarrow$	$\langle 0,0,0\rangle$	$\langle 2,0,0\rangle$	$\langle 0,2,0\rangle$	$\langle 0,0,2\rangle$
$\langle 0,0,0\rangle$	0.978571	0.007143	0.007143	0.007143
$\langle 2,0,0\rangle$	0.135714	0.578571		
$\langle 0,2,0\rangle$	0.135714		0.850000	
$\langle 0,0,2\rangle$	0.135714			0.850000
$\langle 2,2,0\rangle$			0.135714	
$\langle 2,0,2\rangle$				0.135714
$\langle 0,2,2\rangle$			0.135714	0.135714
$\downarrow$from/to$\rightarrow$	$\langle 2,2,0\rangle$	$\langle 2,0,2\rangle$	$\langle 0,2,2\rangle$	$\langle 2,2,2\rangle$
$\langle 2,0,0\rangle$	0.142857	0.142857		
$\langle 0,2,0\rangle$	0.007143		0.007143	
$\langle 0,0,2\rangle$		0.007143	0.007143	
$\langle 2,2,0\rangle$	0.721429			0.142857
$\langle 2,0,2\rangle$		0.721429		0.142857
$\langle 0,2,2\rangle$			0.721429	0.007143
$\langle 2,2,2\rangle$			0.135714	0.864286

Table: $\mathcal{D}_{1,-}\ (\langle reg_1, reg_2, reg_3\rangle)$

the Markov chain $\mathcal{D}$ to compute the *LWA*

$\downarrow$from/to$\rightarrow$	$\langle 0,0,0\rangle$	$\langle 2,0,0\rangle$	$\langle 0,\mathbb{1}\rangle$	$\langle 2,\mathbb{1}\rangle$	$\langle 0,2,2\rangle$	$\langle 2,2,2\rangle$
$\langle 0,0,0\rangle$	0.9786	0.0071	0.0143			
$\langle 2,0,0\rangle$	0.1357	0.5786		0.2857		
$\langle 0,\mathbb{1}\rangle$	0.1357		0.8500	0.0071	0.0071	
$\langle 2,\mathbb{1}\rangle$			0.1357	0.7214		0.1429
$\langle 0,2,2\rangle$			0.2714		0.7214	0.0071
$\langle 2,2,2\rangle$					0.1357	0.8643

Table: $\mathcal{D}'_{1,-}$

the Markov chain $\mathcal{D}$ to compute the *LWA*

↓from/to→	$\langle 0 \rangle$	$\langle 1 \rangle$	$\langle 2 \rangle$
$\langle 0 \rangle$	$r_4 = 0.982972$	$s_4 = 0.008687$	$t_4 = 0.008341$
$\langle 1 \rangle$	$u_4 = 0.055813$	$v_4 = 0.930721$	$w_4 = 0.013466$
$\langle 2 \rangle$	$x_4 = 0.081422$	$y_4 = 0.023461$	$z_4 = 0.895117$

Table: $\mathcal{D}_{\pi_4}$

recomposition

$\mathcal{D}' = zeros(324)$;
for $j = 1 : 6$ **do**
  for $l = 1 : 54$ **do**
    for $i = 1 : 6$ **do**
      for $k = 1 : 54$ **do**
        if $i \neq \wedge l \neq k$ **then**
          $\mathcal{D}'((j-1) \cdot 54 + l, (i-1) \cdot 54 + l) =$
          $\mathcal{D}'_{1,-}((j-1) \cdot 54 + l, (i-1) \cdot 54 + l) + \mathcal{D}'_{1,-}(j,i) \cdot \mathcal{D}'_2(l,k) \cdot \frac{3}{7}$;
          $\mathcal{D}'((j-1) \cdot 54 + l, (j-1) \cdot 54 + k) =$
          $\mathcal{D}'_{1,-}((j-1) \cdot 54 + l, (j-1) \cdot 54 + k) + \mathcal{D}'_{1,-}(j,i) \cdot \mathcal{D}'_2(l,k) \cdot \frac{4}{7}$;
        else
          $\mathcal{D}'((j-1) \cdot 54 + l, (i-1) \cdot 54 + k) =$
          $\mathcal{D}'_{1,-}((j-1) \cdot 54 + l, (i-1) \cdot 54 + k) + \mathcal{D}'_{1,-}(j,i) \cdot \mathcal{D}'_2(l,k)$;

Figure: Recomposition of $\mathcal{D}'_{1,-}$ and $\mathcal{D}'_2$ to $\overline{\mathcal{D}'}$

recomposition

- matrix multiplication is tricky
- serial execution semantics prohibit Kronecker/Hadamard

Equivalence Class Qualifications

Definition 2

$$\begin{aligned} s_i \sim s_j :\Leftrightarrow \quad & ((s_i \models \mathcal{P} \vee s_j \models \mathcal{P}) \wedge \\ & (s_i \not\models \mathcal{P} \vee s_j \not\models \mathcal{P})) \vee \\ & \forall s \in S : prob(s_i, s) = prob(s_j, s) \end{aligned}$$

$\mathcal{D}$ Lumping (strongly probabilistic bisimilar)

Definition 3

$$red(\mathcal{D}, \mathcal{P}) = (\mathcal{D}', \mathcal{P}') \quad (1)$$

$$\mathcal{D}' = (S_{red}, prob_{red}) \quad (2)$$

$$S_{red} = \{[s]_\sim | s \in S\} \quad (3)$$

$$prob_{red}([s_i]_\sim, [s_j]_\sim) = \sum_{d_i \in [s_i]_\sim} prob(d_i, d_j), d_j \in [s_j]_\sim \quad (4)$$

$$[s]_\sim \models \mathcal{P}' :\Leftrightarrow \exists d \in [s]_\sim : d \models \mathcal{P} \quad (5)$$

Theorem 1/2

Theorem 1

$$prob^{\infty}_{red}([s]_{\sim}) = \sum_{d \in [s]_{\sim}} prob^{\infty}(d) \tag{6}$$

Proof

Proof Part 1

Let $prob^0$ be an arbitrary initial distribution for $\mathcal{D}$ and let $prob^0_{red}([s]_\sim) = \sum_{d \in [s]_\sim} prob^0(d)$ be an initial distribution for $\mathcal{D}'$.
Show that for $prob^k$ and $prob^k_{red}$, which are the probability distributions for $\mathcal{D}$ ($\mathcal{D}'$ respectively) at time point k with an initial distribution $prob^0$ ($prob^0_{red}$ respectively) the following holds:

$$\forall k : prob^k_{red}([s]_\sim) = \sum_{d \in [s]_\sim} prob^k(d) \tag{7}$$

Proof per induction over k.
Anchor: $k = 0$ holds by assumption.
Step: show that the following holds

Proof

Proof Part 2

Assumption: $prob_{red}^{k+1}([s]_\sim) =$

$$= \sum_{[d]_\sim \in S_{red}} prob_{red}^{k}([d]_\sim) \cdot prob_{red}([d]_\sim, [s]_\sim)$$

$$= \sum_{[d]_\sim \in S_{red}} (\sum_{e \in [d]_\sim} prob^{k}(e)) \cdot (\sum_{f \in [s]_\sim} prob(d, f))$$

$$= \sum_{[d]_\sim \in S_{red}} \sum_{e \in [d]_\sim} \sum_{f \in [s]_\sim} prob^{k}(e) \cdot prob(d, f)$$

and with $prob(e, f) = prob(d, f)$

$$= \sum_{[d]_\sim \in S_{red}} \sum_{e \in [d]_\sim} \sum_{f \in [s]_\sim} prob^{k}(e) \cdot prob(e, f)$$

$$= \sum_{e \in S} \sum_{f \in [s]_\sim} prob^{k}(e) \cdot prob(e, f)$$

$$= \sum_{f \in [s]_\sim} \sum_{e \in S} prob^{k}(e) \cdot prob(e, f)$$

$$= \sum_{d \in [s]_\sim} prob^{k+1}(d)$$

Proof

Proof Part 3

Thereby, $\forall k : prob^{k}_{red}([s]_{\sim}) = \sum_{d \in [s]_{\sim}} prob^{k}(d).$ □

Corollary

Corollary 1

Theorem 1 and the first two conditions from Definition 2 imply that the limiting availability LWA_0 *satisfies* $LWA_0(\mathcal{D}, \mathcal{P}) = LWA_0(\mathcal{D}', \mathcal{P}')$. *Thereby* $LWA_0(\mathcal{D}, \mathcal{P}) = \sum_{s \models \mathcal{P}} prob^{\infty}(s)$ *and consequently* $LWA_0(\mathcal{D}', \mathcal{P}') = \sum_{[s]_\sim \models \mathcal{P}'} prob^{\infty}_{red}([s]_\sim)$.

Theorem 2/2

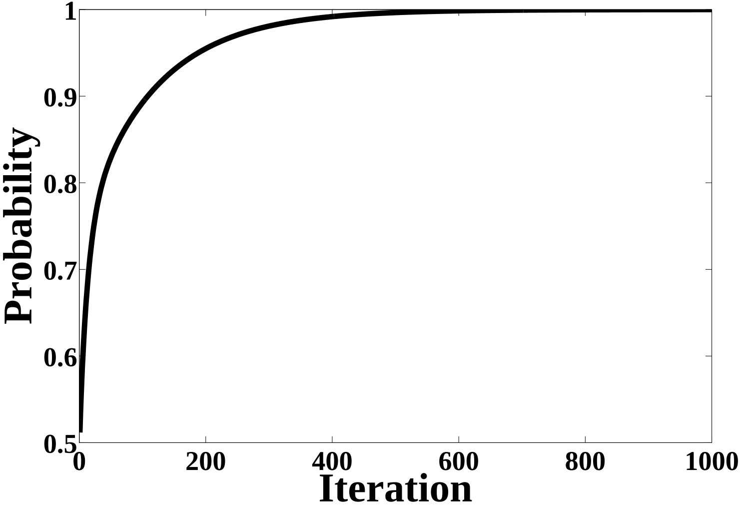
Theorem 2

$LWA(\mathcal{D},\mathcal{P}) \sim LWA(\mathcal{D}',\mathcal{P}') : red(\mathcal{D},\mathcal{P}) = (\mathcal{D}',\mathcal{P}')$
with $\mathcal{D} = (S, prob), prob : S \times S \to \mathbb{R}$

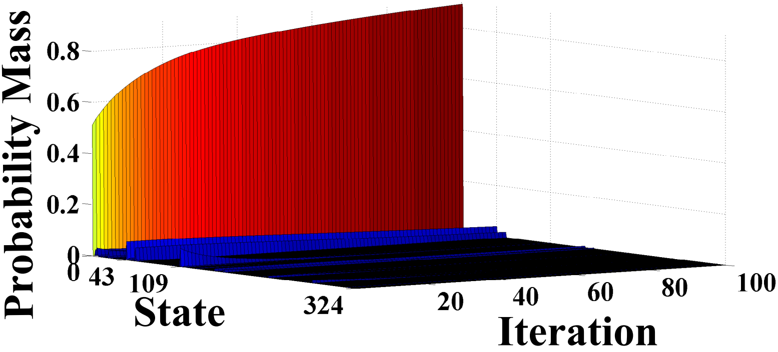
Proof 2/2

analogously...

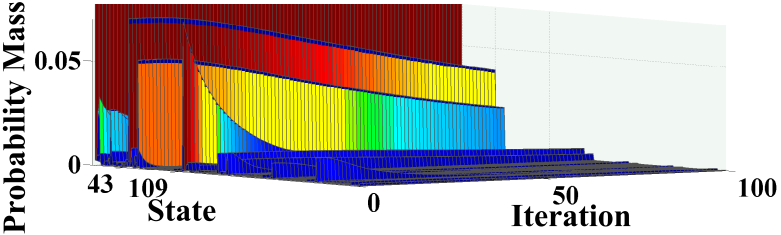
So what?



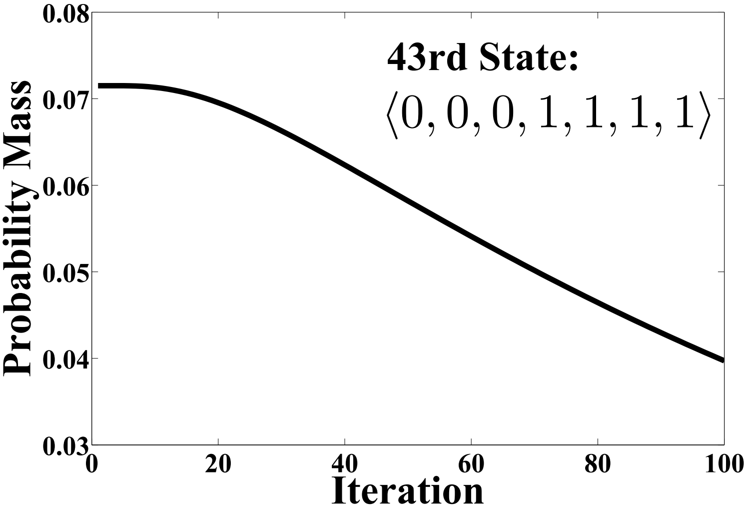
So what?


Probability Mass
0.8
0.6
0.4
0.2
0
0
43
109
324
State
20
40
60
80
100
Iteration

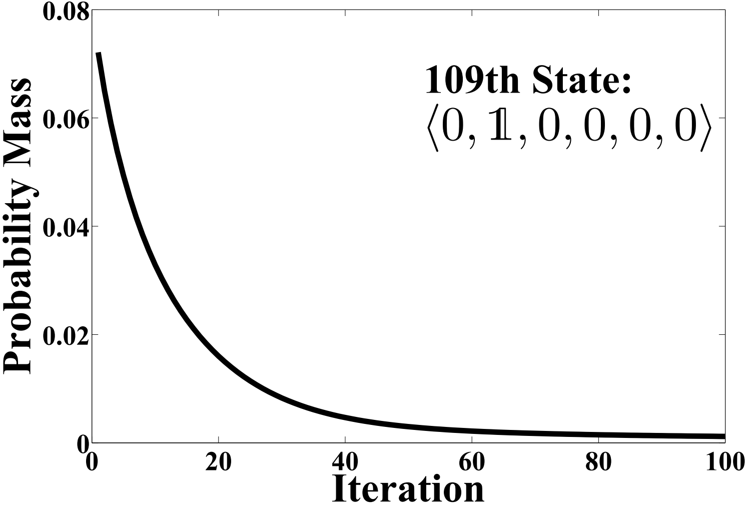
So what?



So what?


43rd State:
$\langle 0, 0, 0, 1, 1, 1, 1 \rangle$
Probability Mass
Iteration
0.08
0.07
0.06
0.05
0.04
0.03
0
20
40
60
80
100

So what?


0.08
0.06
0.04
0.02
0
Probability Mass
0
20
40
60
80
100
Iteration
109th State:
$\langle 0, \mathbb{1}, 0, 0, 0, 0 \rangle$

Combining Decomposition and Reduction for State Space Analysis of a Self-Stabilizing System

Nils Müllner

nils.muellner@informatik.uni-oldenburg.de
Abteilung Systemsoftware und verteilte Systeme
Department für Informatik
Carl von Ossietzky Universität Oldenburg

29. March 2012