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
Example

Decomposition and Reduction: Example

Computation of Limiting Window Availability

Conclusion

Outline

1. Example

2. Decomposition and Reduction: Example

3. Computation of Limiting Window Availability

4. Conclusion
self-stabilizing broadcast

const id := 0,
var reg,
repeat{
  reg := 0}

Figure: Root Process

const neighbors := \{π_i, \ldots\},
const id := \min\{id(π_i), \ldots\} + 1,
var reg,
var set := reg_i, π(reg_i) \in
neighbors | \forall i : id(π_i) = id − 1
repeat{
  \neg ((\exists reg_i : π(reg_i) \in set \land reg_i = 2)
\ xor (\exists reg_i : π(reg_i) \in set \land reg_i = 0))
  → reg := 1
□∃ reg_i : π(reg_i) \in set \land reg_i = 0
  → reg := 0
□∃ reg_i : π(reg_i) \in set \land reg_i = 2
  → reg := 2}

Figure: Broadcast Sub-Algorithm for Non-Root Processes
self-stabilizing broadcast

\texttt{const \hspace{0.5em} id := 0,}
\texttt{var \hspace{0.5em} reg,}
\texttt{repeat}\{
\texttt{\hspace{1em}reg := 0}
\}\texttt{

Figure: Root Process

\texttt{const \hspace{0.5em} neighbors := \{\pi_i, \ldots\},}
\texttt{const \hspace{0.5em} id := \min\{id(\pi_i), \ldots\} + 1,}
\texttt{var \hspace{0.5em} reg,}
\texttt{var \hspace{0.5em} set := reg_i, \pi(reg_i) \in neighbors | \forall i : id(\pi_i) = id - 1}
\texttt{repeat}\{
\texttt{\hspace{1em}\neg((\exists \hspace{0.5em}reg_i : \pi(reg_i) \in set \land reg_i = 2)}
\texttt{xor(\exists \hspace{0.5em}reg_i : \pi(reg_i) \in set \land reg_i = 0))}
\texttt{\rightarrow \hspace{0.5em}reg := 1}
\texttt{\Box \exists \hspace{0.5em}reg_i : \pi(reg_i) \in set \land reg_i = 0}
\texttt{\rightarrow \hspace{0.5em}reg := 0}
\texttt{\Box \exists \hspace{0.5em}reg_i : \pi(reg_i) \in set \land reg_i = 2}
\texttt{\rightarrow \hspace{0.5em}reg := 2}
\}\texttt{

Figure: Broadcast Sub-Algorithm for Non-Root Processes}
self-stabilizing broadcast

\begin{align*}
\text{const } & \text{id} := 0, \\
\text{var } & \text{reg}, \\
\text{repeat} \\
& \text{reg} := 0
\end{align*}

\textbf{Figure: Root Process}

\begin{align*}
\text{const } & \text{neighbors} := \{\pi_i, \ldots\}, \\
\text{const } & \text{id} := \min \{id(\pi_i), \ldots\} + 1, \\
\text{var } & \text{reg}, \\
\text{var } & \text{set} := \text{reg}_i, \pi(\text{reg}_i) \in \\
& \text{neighbors} | \forall i : id(\pi_i) = id - 1
\end{align*}

\text{repeat} \\
& \neg((\exists \text{reg}_i : \pi(\text{reg}_i) \in \text{set} \land \text{reg}_i = 2) \\
& \hspace{1cm} \text{xor}(\exists \text{reg}_i : \pi(\text{reg}_i) \in \text{set} \land \text{reg}_i = 0)) \\
& \hspace{1cm} \rightarrow \text{reg} := 1 \\
& \hspace{1cm} \Box \exists \text{reg}_i : \pi(\text{reg}_i) \in \text{set} \land \text{reg}_i = 0 \\
& \hspace{1cm} \rightarrow \text{reg} := 0 \\
& \hspace{1cm} \Box \exists \text{reg}_i : \pi(\text{reg}_i) \in \text{set} \land \text{reg}_i = 2 \\
& \hspace{1cm} \rightarrow \text{reg} := 2

\textbf{Figure: Broadcast Sub-Algorithm for Non-Root Processes}
self-stabilizing broadcast

```plaintext
const id := 0,
var reg,
repeat{
  reg := 0
}
```

**Figure: Root Process**

```
const neighbors := \{\pi_i, \ldots\},
const id := \min\{id(\pi_i), \ldots\} + 1,
var reg,
var set := reg_i, \pi(reg_i) \in neighbors | \forall i : id(\pi_i) = id - 1
repeat{
  ¬((\exists reg_i : \pi(reg_i) \in set \land reg_i = 2) \quad xor (\exists reg_i : \pi(reg_i) \in set \land reg_i = 0))
  → reg := 1
  □\exists reg_i : \pi(reg_i) \in set \land reg_i = 0
  → reg := 0
  □\exists reg_i : \pi(reg_i) \in set \land reg_i = 2
  → reg := 2
```

**Figure: Broadcast Sub-Algorithm for Non-Root Processes**
self-stabilizing broadcast

\[
\begin{align*}
\text{const } & \text{id} := 0, \\
\text{var } & \text{reg,} \\
\text{repeat}\{ & \text{reg} := 0
\}
\end{align*}
\]

\textbf{Figure: Root Process}

\[
\begin{align*}
\text{id} &= i \\
\text{id} &= 0 \\
\text{id} &= i + 1
\end{align*}
\]

\[
\begin{align*}
\text{const } & \text{neighbors} := \{\pi_i, \ldots\}, \\
\text{const } & \text{id} := \min\{\text{id}(\pi_i), \ldots\} + 1, \\
\text{var } & \text{reg,} \\
\text{var } & \text{set} := \text{reg}_i, \pi(\text{reg}_i) \in \text{neighbors} \mid \forall i : \text{id}(\pi_i) = \text{id} - 1 \\
\text{repeat}\{ & \\
\neg((\exists \text{reg}_i : \pi(\text{reg}_i) \in \text{set} \land \text{reg}_i = 2) \\
\xor(\exists \text{reg}_i : \pi(\text{reg}_i) \in \text{set} \land \text{reg}_i = 0)) & \rightarrow \text{reg} := 1 \\
\Box \exists \text{reg}_i : \pi(\text{reg}_i) \in \text{set} \land \text{reg}_i = 0 & \rightarrow \text{reg} := 0 \\
\Box \exists \text{reg}_i : \pi(\text{reg}_i) \in \text{set} \land \text{reg}_i = 2 & \rightarrow \text{reg} := 2
\}
\]

\textbf{Figure: Broadcast Sub-Algorithm for Non-Root Processes}

Nils Müller

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

\[
\text{const } \text{id} := 0, \\
\text{var } \text{reg}, \\
\text{repeat}\{ \\
\text{reg} := 0 \}
\]

\textbf{Figure: Root Process}

\[
\text{const } \text{neighbors} := \{\pi_i, \ldots\}, \\
\text{const } \text{id} := \min\{\text{id}(\pi_i), \ldots\} + 1, \\
\text{var } \text{reg}, \\
\text{var } \text{set} := \text{reg}_i, \pi(\text{reg}_i) \in \\
\text{neighbors}\mid \forall i: \text{id}(\pi_i) = \text{id} - 1
\]

\text{repeat}\{
\neg((\exists \text{reg}_i: \pi(\text{reg}_i) \in \text{set} \land \text{reg}_i = 2) \\
\lor(\exists \text{reg}_i: \pi(\text{reg}_i) \in \text{set} \land \text{reg}_i = 0)) \\
\rightarrow \text{reg} := 1 \\
\Box \exists \text{reg}_i: \pi(\text{reg}_i) \in \text{set} \land \text{reg}_i = 0 \\
\rightarrow \text{reg} := 0 \\
\Box \exists \text{reg}_i: \pi(\text{reg}_i) \in \text{set} \land \text{reg}_i = 2 \\
\rightarrow \text{reg} := 2
\}

\textbf{Figure: Broadcast Sub-Algorithm for Non-Root Processes}
self-stabilizing broadcast

const id := 0,
var reg,
repeat{
  reg := 0
}

**Figure:** Root Process

const neighbors := \{\pi_i, \ldots\},
const id := \min\{id(\pi_i), \ldots\} + 1,
var reg,
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 \land reg_i = 2)
  \ xor(\exists reg_i : \pi(reg_i) \in set \land reg_i = 0))
  \rightarrow reg := 1
\square \exists reg_i : \pi(reg_i) \in set \land reg_i = 0
  \rightarrow reg := 0
\square \exists reg_i : \pi(reg_i) \in set \land reg_i = 2
  \rightarrow reg := 2
}

**Figure:** Broadcast Sub-Algorithm for Non-Root Processes
self-stabilizing broadcast

\[
\begin{align*}
\text{const } id & := 0, \\
\text{var } reg, \\
\text{repeat} \\
& \quad \text{reg } := 0
\end{align*}
\]

\textit{Figure: Root Process}

\[
\begin{align*}
\text{const } neighbors & := \{\pi_i, \ldots\}, \\
\text{const } id & := \min\{id(\pi_i), \ldots\} + 1, \\
\text{var } reg, \\
\text{var } set := \text{reg}_i, \pi(\text{reg}_i) \in \\
& \quad \text{neighbors} | \forall i : id(\pi_i) = id - 1 \\
\text{repeat} \\
& \quad \neg((\exists \text{reg}_i : \pi(\text{reg}_i) \in set \land \text{reg}_i = 2) \\
& \quad \quad \lor (\exists \text{reg}_i : \pi(\text{reg}_i) \in set \land \text{reg}_i = 0)) \\
& \quad \quad \rightarrow \text{reg } := 1 \\
\quad \Box \exists \text{reg}_i : \pi(\text{reg}_i) \in set \land \text{reg}_i = 0 \\
& \quad \quad \rightarrow \text{reg } := 0 \\
\quad \Box \exists \text{reg}_i : \pi(\text{reg}_i) \in set \land \text{reg}_i = 2 \\
& \quad \quad \rightarrow \text{reg } := 2
\end{align*}
\]

\textit{Figure: Broadcast Sub-Algorithm for Non-Root Processes}
self-stabilizing broadcast

\[
\text{const } id := 0, \\
\text{var } reg, \\
\text{repeat}
\{
    \text{reg} := 0
\}
\]

Figure: Root Process

\[
\begin{align*}
\pi_1 & \rightarrow \pi_2 \\
\pi_1 & \rightarrow \pi_4 \\
\pi_4 & \rightarrow \pi_5 \\
\pi_4 & \rightarrow \pi_7 \\
\pi_2 & \rightarrow \pi_3 \\
\pi_2 & \rightarrow \pi_6
\end{align*}
\]

\[s_t \models P : \text{reg}_1 = 0 \land \ldots \land \text{reg}_7 = 0\]

\[
\text{const } \text{neighbors} := \{\pi_i, \ldots\}, \\
\text{const } id := \min\{id(\pi_i), \ldots\} + 1, \\
\text{var } \text{reg}, \\
\text{var } \text{set} := \text{reg}_i, \pi(\text{reg}_i) \in \text{neighbors}\mid \forall i : id(\pi_i) = id - 1
\]

\[
\text{repeat}
\{
    \neg((\exists \text{reg}_i : \pi(\text{reg}_i) \in \text{set} \land \text{reg}_i = 2)
    \lor (\exists \text{reg}_i : \pi(\text{reg}_i) \in \text{set} \land \text{reg}_i = 0))
    \rightarrow \text{reg} := 1
\]

\[
\Box \exists \text{reg}_i : \pi(\text{reg}_i) \in \text{set} \land \text{reg}_i = 0
    \rightarrow \text{reg} := 0
\]

\[
\Box \exists \text{reg}_i : \pi(\text{reg}_i) \in \text{set} \land \text{reg}_i = 2
    \rightarrow \text{reg} := 2
\]

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:

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

$w$ is called window size.
Trivia

- has 648 states: \(2 \times 2 \times 2 \times 3 \times 3 \times 3 \times 3\),
  \(\langle 0,0,0,0,0,0,0 \rangle, \ldots, \langle 2,2,2,2,2,2,2 \rangle\)
- probabilistic scheduler
- serial execution semantics (to exclude hazards)
- transient faults \(q = 0.05\)
- processes \([\pi_2 \text{ with } \pi_3]\), and \([\pi_5 \text{ with } \pi_6]\), are strongly probabilistic bisimilar
- lumping allows for reduction to 324 states
Plan of Action

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

\[ D_1, - \]

\[ D_{\pi_4} \]
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

<table>
<thead>
<tr>
<th>from/to</th>
<th>$\langle 0, 0, 0 \rangle$</th>
<th>$\langle 2, 0, 0 \rangle$</th>
<th>$\langle 0, 2, 0 \rangle$</th>
<th>$\langle 0, 0, 2 \rangle$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\langle 0, 0, 0 \rangle$</td>
<td>0.978571</td>
<td>0.007143</td>
<td>0.007143</td>
<td>0.007143</td>
</tr>
<tr>
<td>$\langle 2, 0, 0 \rangle$</td>
<td>0.135714</td>
<td>0.578571</td>
<td>0.850000</td>
<td>0.850000</td>
</tr>
<tr>
<td>$\langle 0, 2, 0 \rangle$</td>
<td>0.135714</td>
<td>0.135714</td>
<td>0.135714</td>
<td>0.135714</td>
</tr>
<tr>
<td>$\langle 0, 0, 2 \rangle$</td>
<td>0.135714</td>
<td>0.135714</td>
<td>0.135714</td>
<td>0.135714</td>
</tr>
<tr>
<td>$\langle 2, 2, 0 \rangle$</td>
<td>0.135714</td>
<td>0.135714</td>
<td>0.135714</td>
<td>0.135714</td>
</tr>
<tr>
<td>$\langle 2, 0, 2 \rangle$</td>
<td>0.135714</td>
<td>0.135714</td>
<td>0.135714</td>
<td>0.135714</td>
</tr>
<tr>
<td>$\langle 0, 2, 2 \rangle$</td>
<td>0.135714</td>
<td>0.135714</td>
<td>0.135714</td>
<td>0.135714</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>from/to</th>
<th>$\langle 2, 2, 0 \rangle$</th>
<th>$\langle 2, 0, 2 \rangle$</th>
<th>$\langle 0, 2, 2 \rangle$</th>
<th>$\langle 2, 2, 2 \rangle$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\langle 2, 0, 0 \rangle$</td>
<td>0.142857</td>
<td>0.142857</td>
<td>0.142857</td>
<td>0.142857</td>
</tr>
<tr>
<td>$\langle 0, 2, 0 \rangle$</td>
<td>0.007143</td>
<td>0.007143</td>
<td>0.007143</td>
<td>0.007143</td>
</tr>
<tr>
<td>$\langle 0, 0, 2 \rangle$</td>
<td>0.007143</td>
<td>0.007143</td>
<td>0.007143</td>
<td>0.007143</td>
</tr>
<tr>
<td>$\langle 2, 2, 0 \rangle$</td>
<td>0.721429</td>
<td>0.721429</td>
<td>0.721429</td>
<td>0.721429</td>
</tr>
<tr>
<td>$\langle 2, 0, 2 \rangle$</td>
<td>0.721429</td>
<td>0.721429</td>
<td>0.721429</td>
<td>0.721429</td>
</tr>
<tr>
<td>$\langle 0, 2, 2 \rangle$</td>
<td>0.721429</td>
<td>0.721429</td>
<td>0.721429</td>
<td>0.721429</td>
</tr>
<tr>
<td>$\langle 2, 2, 2 \rangle$</td>
<td>0.721429</td>
<td>0.721429</td>
<td>0.721429</td>
<td>0.721429</td>
</tr>
</tbody>
</table>

Table: $\mathcal{D}_{1,-} \left( \langle \text{reg}_1, \text{reg}_2, \text{reg}_3 \rangle \right)$
the Markov chain $\mathcal{D}$ to compute the $LWA$

| $\downarrow$from/to$
\rightarrow$ | $\langle 0, 0, 0 \rangle$ | $\langle 2, 0, 0 \rangle$ | $\langle 0, 1 \rangle$ | $\langle 2, 1 \rangle$ | $\langle 0, 2, 2 \rangle$ | $\langle 2, 2, 2 \rangle$ |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\langle 0, 0, 0 \rangle$</td>
<td>0.9786</td>
<td>0.0071</td>
<td>0.0143</td>
<td>0.2857</td>
<td>0.0071</td>
<td>0.1357</td>
</tr>
<tr>
<td>$\langle 2, 0, 0 \rangle$</td>
<td>0.1357</td>
<td>0.5786</td>
<td>0.8500</td>
<td>0.0071</td>
<td>0.7214</td>
<td>0.0071</td>
</tr>
<tr>
<td>$\langle 0, 1 \rangle$</td>
<td>0.1357</td>
<td>0.0143</td>
<td>0.8500</td>
<td>0.0071</td>
<td>0.7214</td>
<td>0.0071</td>
</tr>
<tr>
<td>$\langle 2, 1 \rangle$</td>
<td>0.2714</td>
<td>0.0071</td>
<td>0.7214</td>
<td>0.0071</td>
<td>0.8643</td>
<td></td>
</tr>
<tr>
<td>$\langle 0, 2, 2 \rangle$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\langle 2, 2, 2 \rangle$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table: $\mathcal{D}'_{1,-}$
the Markov chain $\mathcal{D}$ to compute the LWA

<table>
<thead>
<tr>
<th>↓from/to→</th>
<th>$\langle 0 \rangle$</th>
<th>$\langle 1 \rangle$</th>
<th>$\langle 2 \rangle$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\langle 0 \rangle$</td>
<td>$r_4 = 0.982972$</td>
<td>$s_4 = 0.008687$</td>
<td>$t_4 = 0.008341$</td>
</tr>
<tr>
<td>$\langle 1 \rangle$</td>
<td>$u_4 = 0.055813$</td>
<td>$v_4 = 0.930721$</td>
<td>$w_4 = 0.013466$</td>
</tr>
<tr>
<td>$\langle 2 \rangle$</td>
<td>$x_4 = 0.081422$</td>
<td>$y_4 = 0.023461$</td>
<td>$z_4 = 0.895117$</td>
</tr>
</tbody>
</table>

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

\[ D' = \text{zeros}(324); \]
\[ \text{for } j = 1 : 6 \text{ do} \]
\[ \quad \text{for } l = 1 : 54 \text{ do} \]
\[ \quad \quad \text{for } i = 1 : 6 \text{ do} \]
\[ \quad \quad \quad \text{for } k = 1 : 54 \text{ do} \]
\[ \quad \quad \quad \quad \text{if } i \neq j \land l \neq k \text{ then} \]
\[ \quad \quad \quad \quad \quad D'((j - 1) \cdot 54 + l, (i - 1) \cdot 54 + l) = \]
\[ \quad \quad \quad \quad \quad D'_{1, -}(j - 1) \cdot 54 + l, (i - 1) \cdot 54 + l) + D'_{1, -}(j, i) \cdot D'_{2}(l, k) \cdot \frac{3}{7}; \]
\[ \quad \quad \quad \quad \quad D'((j - 1) \cdot 54 + l, (j - 1) \cdot 54 + k) = \]
\[ \quad \quad \quad \quad \quad D'_{1, -}(j - 1) \cdot 54 + l, (j - 1) \cdot 54 + k) + D'_{1, -}(j, i) \cdot D'_{2}(l, k) \cdot \frac{4}{7}; \]
\[ \quad \quad \quad \quad \text{else} \]
\[ \quad \quad \quad \quad \quad D'((j - 1) \cdot 54 + l, (i - 1) \cdot 54 + k) = \]
\[ \quad \quad \quad \quad \quad D'_{1, -}(j - 1) \cdot 54 + l, (i - 1) \cdot 54 + k) + D'_{1, -}(j, i) \cdot D'_{2}(l, k); \]

**Figure:** Recomposition of \( D'_{1, -} \) and \( D'_{2} \) to \( \overline{D'} \)
recomposition

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

Definition 2

\( s_i \sim s_j : \iff ((s_i \models P \lor s_j \models P) \land \\
(s_i \not\models P \lor s_j \not\models P)) \lor \\
\forall s \in S : \text{prob}(s_i, s) = \text{prob}(s_j, s) \)
\(D\) Lumping (strongly probabilistic bisimilar)

**Definition 3**

\[
\text{red}(D, P) = (D', P') \quad (1)
\]

\[
D' = (S_{\text{red}}, \text{prob}_{\text{red}}) \quad (2)
\]

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

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

\[
[s]_\sim \models P' : \iff \exists d \in [s]_\sim : d \models P \quad (5)
\]
Theorem 1/2

Theorem 1

\[ \text{prob}_{\text{red}}^\infty([s]) = \sum_{d \in [s]} \text{prob}^\infty(d) \]  \hspace{1cm} (6)
Proof

Proof Part 1

Let \( \text{prob}^0 \) be an arbitrary initial distribution for \( \mathcal{D} \) and let

\[
\text{prob}^0_{\text{red}}([s]_\sim) = \sum_{d \in [s]_\sim} \text{prob}^0(d)
\]

be an initial distribution for \( \mathcal{D}' \).

Show that for \( \text{prob}^k \) and \( \text{prob}^k_{\text{red}} \), which are the probability distributions for \( \mathcal{D} \) (\( \mathcal{D}' \) respectively) at time point \( k \) with an initial distribution \( \text{prob}^0 \) (\( \text{prob}^0_{\text{red}} \) respectively) the following holds:

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

Proof per induction over \( k \).

Anchor: \( k = 0 \) holds by assumption.

Step: show that the following holds
Proof

Proof Part 2

**Assumption:** \( \text{prob}^{k+1}_{\text{red}}([s]_{\sim}) = \)

\[
= \sum_{[d]_{\sim} \in S_{\text{red}}} \text{prob}^k_{\text{red}}([d]_{\sim}) \cdot \text{prob}_{\text{red}}([d]_{\sim}, [s]_{\sim})
\]

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

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

**and with** \( \text{prob}(e, f) = \text{prob}(d, f) \)

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

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

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

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

Proof Part 3

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

$\Box$
Corollary

Corollary 1

*Theorem 1 and the first two conditions from Definition 2 imply that the limiting availability $LWA_0$ satisfies*

$LWA_0(D, P) = LWA_0(D', P')$. *Thereby*

$LWA_0(D, P) = \sum_{s \models P} \text{prob}^{\infty}(s)$ and consequently

$LWA_0(D', P') = \sum_{[s] \sim \models P'} \text{prob}_{\text{red}}^{\infty}([s]_{\sim})$.
Theorem 2/2

**Theorem 2**

\[ \text{LWA}(\mathcal{D}, \mathcal{P}) \sim \text{LWA}(\mathcal{D}', \mathcal{P}') : \text{red}(\mathcal{D}, \mathcal{P}) = (\mathcal{D}', \mathcal{P}') \]

with \( \mathcal{D} = (S, \text{prob}) \), \( \text{prob} : S \times S \rightarrow \mathbb{R} \)
Proof 2/2

analogously...
So what?

![Graph showing the probability of an event as a function of iteration. The x-axis represents iteration, ranging from 0 to 1000, and the y-axis represents probability, ranging from 0.5 to 1. The graph shows a curve that approaches 1 as iteration increases.](image-url)
So what?
So what?
So what?

43rd State: \(\langle 0, 0, 0, 1, 1, 1, 1, 1 \rangle\)
So what?

109th State: 
\[ \langle 0, 1, 0, 0, 0, 0, 0 \rangle \]
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