Parallel Construction of Simultaneous Deterministic Finite Automata on Shared-memory Multicores

Minyoung Jung[1], Jinwoo Park[1],
Johann Blieberger[2] and Bernd Burgstaller[1]
[1]Yonsei University, Korea
[2]Vienna University of Technology, Austria

46th International Conference of Parallel Processing
Bristol, United Kingdom in August 14 - 17, 2017

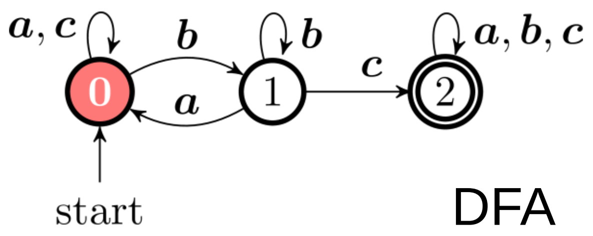
Motivation

- String pattern matching with finite automata (FAs) is a well-established method across many areas.
  - Text editors
  - Compiler front-ends
  - Internet search engines
  - Security and DNA sequence analysis
- The sequential FA algorithm has linear complexity in the size of the input.
  - Significant research effort has been spent on parallelizing FA matching to improve the sequential performance
  - → Hard to be parallelized due to the dependency between state transitions

Motivation (cont.)

- Limitation of parallel FA matching

δ	$\boldsymbol{a}$	$\boldsymbol{b}$	$\boldsymbol{c}$
0	0	1	0
1	0	1	2
2	2	2	2



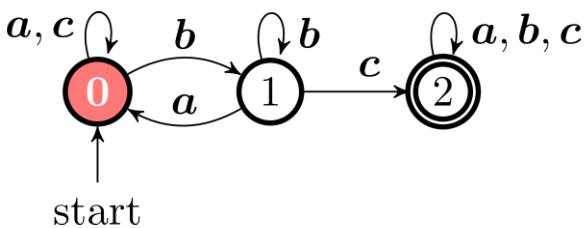
Input: $\boldsymbol{b\,b\,a\,c\,a\,a\,b\,c\,c}$ 9 steps

start **0** $\rightarrow p_0$: 1 1 0 0 0 0 1 2 2

Motivation (cont.)

- Limitation of parallel FA matching

δ	$\boldsymbol{a}$	$\boldsymbol{b}$	$\boldsymbol{c}$
0	0	1	0
1	0	1	2
2	2	2	2



Input: $\boldsymbol{b\,b\,a\,c\,a\,a\,b\,c\,c}$ 9 steps

start $\mathbf{0} \rightarrow p_0$: 1 1 0 0 0 0 1 2 2

$\mathcal{O}(\frac{n}{|P|})$

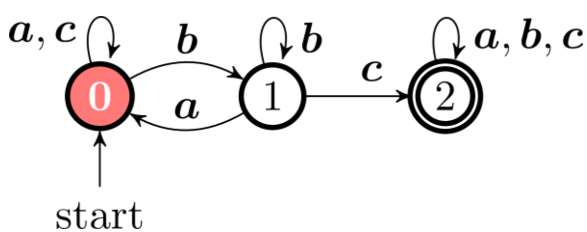
chunks: c_0 $\boldsymbol{b\,b\,a}$ c_1 $\boldsymbol{c\,a\,a}$ c_2 $\boldsymbol{b\,c\,c}$

3 steps 3 steps 3 steps

Motivation (cont.)

- Limitation of parallel FA matching

δ	$\boldsymbol{a}$	$\boldsymbol{b}$	$\boldsymbol{c}$
0	0	1	0
1	0	1	2
2	2	2	2



Input: **b b a c a a b c c** 9 steps

start **0** $\rightarrow p_0$: 1 1 0 0 0 0 1 2 2

$\mathcal{O}(\frac{n}{|P|})$

chunks: c_0 **b b a** c_1 **c a a** c_2 **b c c**

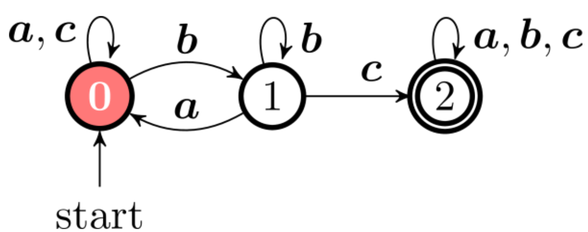
start **0** $\rightarrow p_0$: 1 1 0

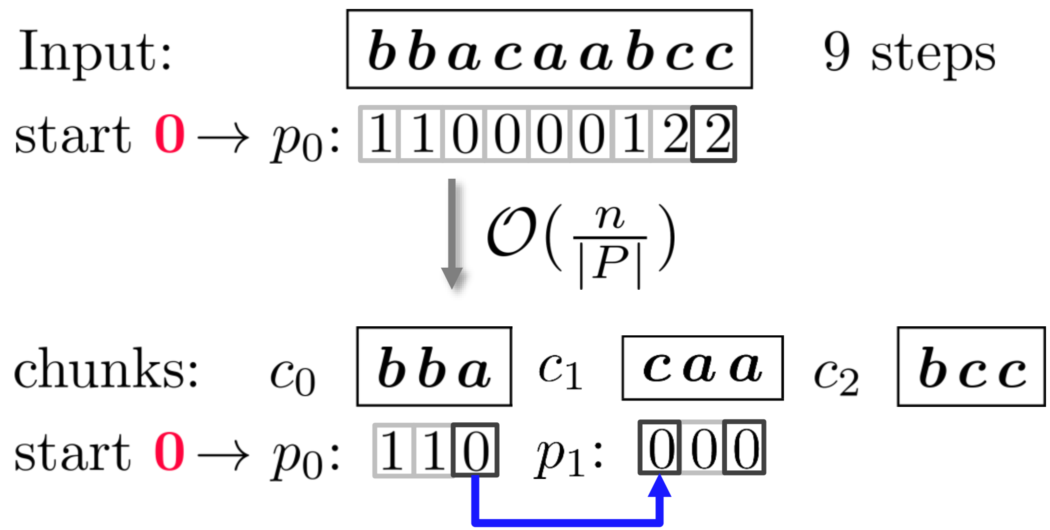
What is the start state?

Motivation (cont.)

- Limitation of parallel FA matching

δ	$\boldsymbol{a}$	$\boldsymbol{b}$	$\boldsymbol{c}$
0	0	1	0
1	0	1	2
2	2	2	2

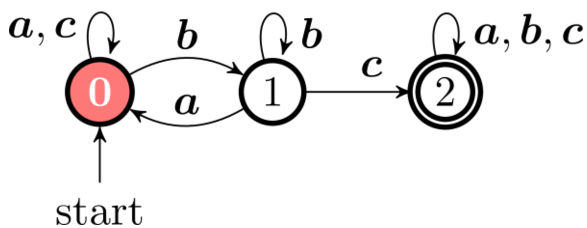


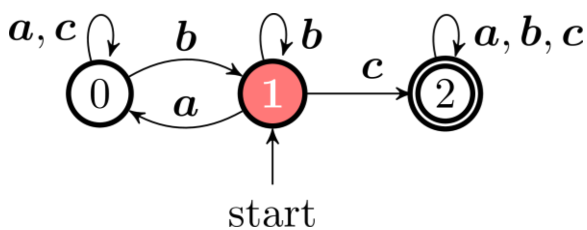


Motivation (cont.)

- Limitation of parallel FA matching

δ	a	b	c
0	0	1	0
1	0	1	2
2	2	2	2





Input: **b b a c a a b c c** 9 steps

start **0** $\rightarrow$ p_0: 1 1 0 0 0 0 1 2 2

$\mathcal{O}(\frac{n}{|P|})$

chunks: c_0 **b b a** c_1 **c a a** c_2 **b c c**

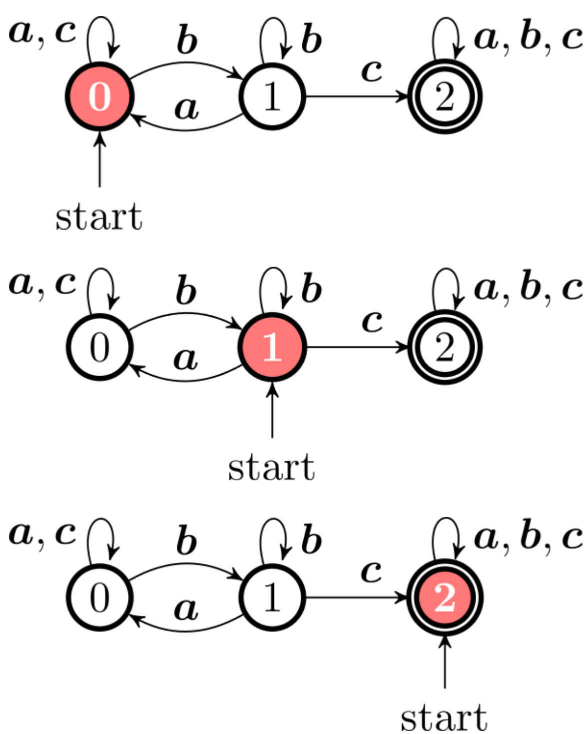
start **0** $\rightarrow$ p_0: 1 1 0 p_1: 0 0 0

start **1** $\rightarrow$ 2 2 2

Motivation (cont.)

- Limitation of parallel FA matching

δ	a	b	c
0	0	1	0
1	0	1	2
2	2	2	2



Input: $\boldsymbol{b\,b\,a\,c\,a\,a\,b\,c\,c}$ 9 steps

start **0** $\rightarrow p_0$: 1 1 0 0 0 0 1 2 2

$\mathcal{O}(\frac{n}{|P|})$

chunks: c_0 $\boldsymbol{b\,b\,a}$ c_1 $\boldsymbol{c\,a\,a}$ c_2 $\boldsymbol{b\,c\,c}$

start **0** $\rightarrow p_0$: 1 1 0 p_1: 0 0 0

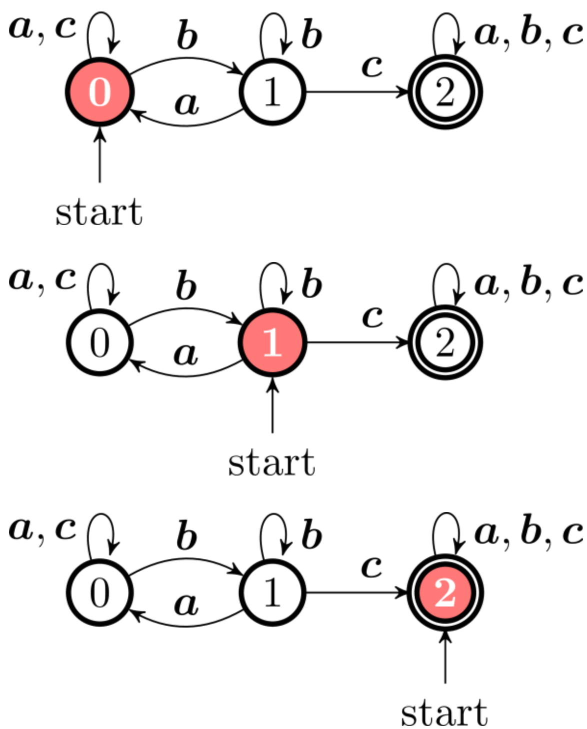
start **1** $\rightarrow$ 2 2 2

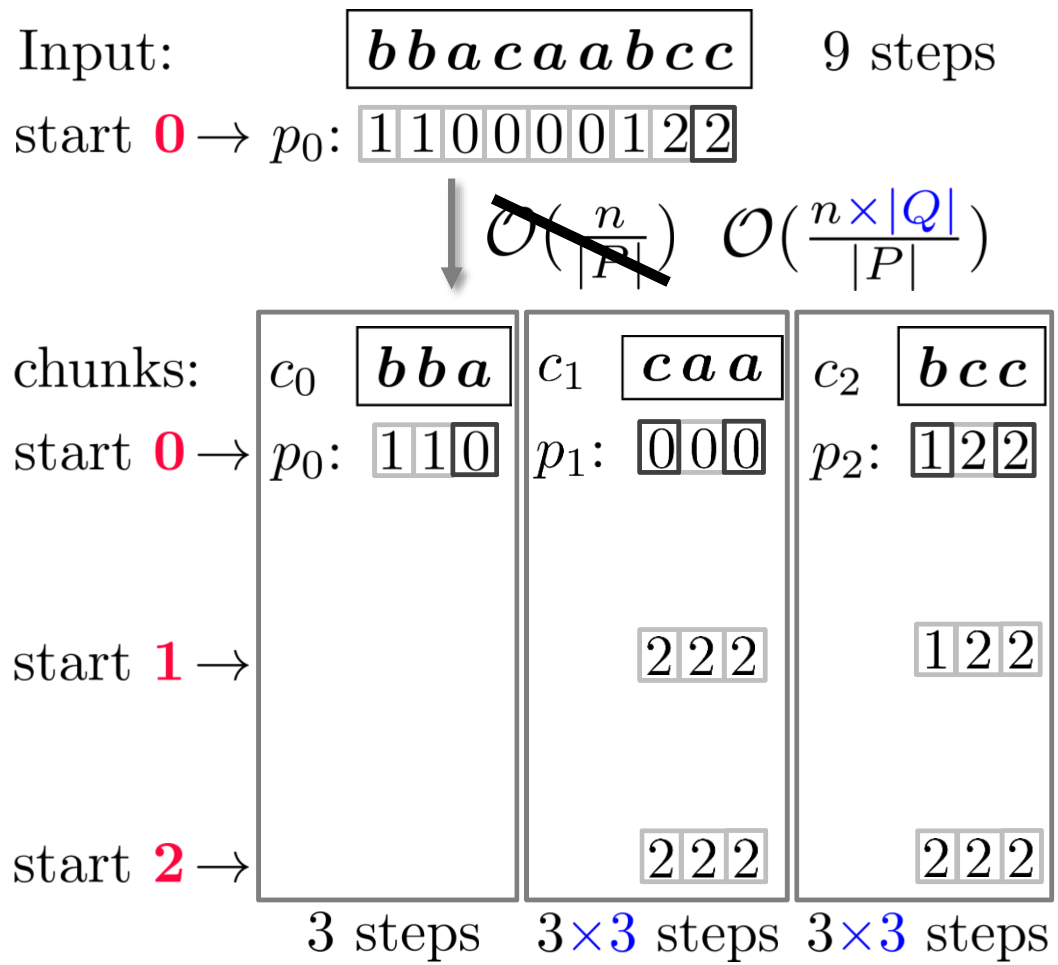
start **2** $\rightarrow$ 2 2 2

Motivation (cont.)

- Limitation of parallel FA matching

δ	a	b	c
0	0	1	0
1	0	1	2
2	2	2	2

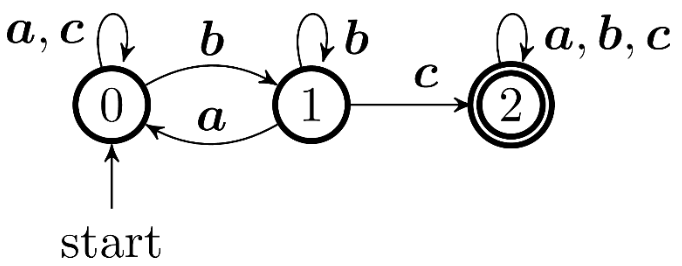




Motivation (cont.)

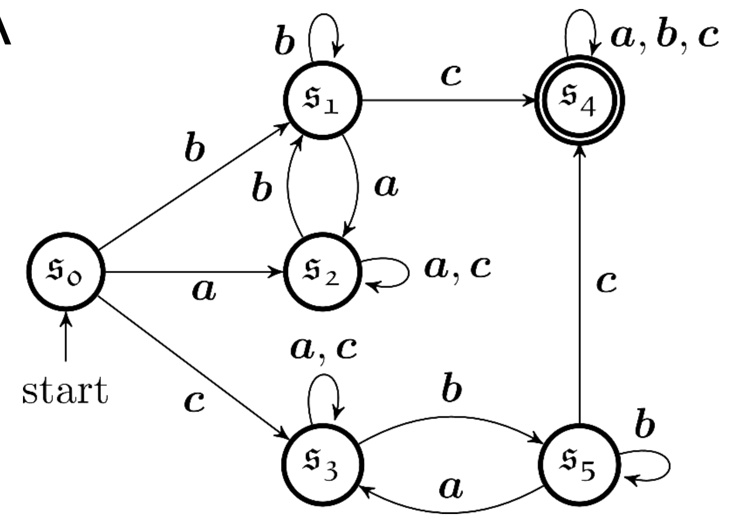
- Simultaneous Finite Automata (SFAs)
  - Accumulated state transition information
  - Simulates the parallel execution of |Q| DFAs on a single DFA

DFA



SFA construction

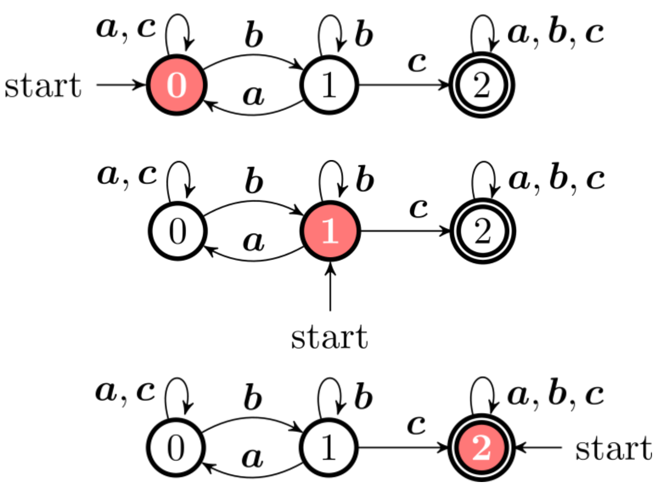
SFA



$\mathfrak{s}_0$	$\langle 0, 1, 2 \rangle$	$\mathfrak{s}_1$	$\langle 1, 1, 2 \rangle$	$\mathfrak{s}_2$	$\langle 0, 0, 2 \rangle$
$\mathfrak{s}_3$	$\langle 0, 2, 2 \rangle$	$\mathfrak{s}_4$	$\langle 2, 2, 2 \rangle$	$\mathfrak{s}_5$	$\langle 1, 2, 2 \rangle$

Motivation (cont.)

- Parallel FA matching



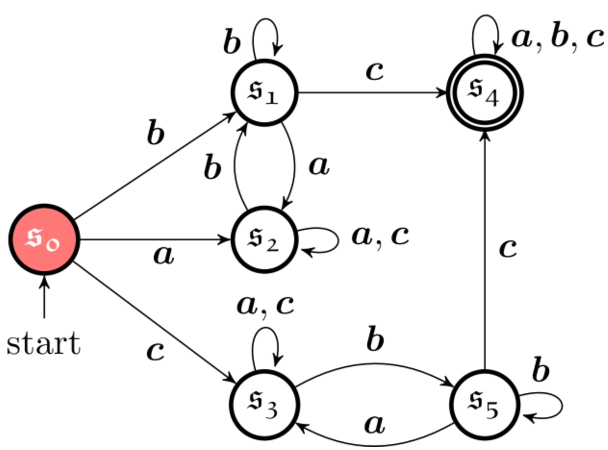
chunks: c_0 **b b a** c_1 **c a a** c_2 **b c c**

start **0** $\rightarrow$

start **1** $\rightarrow$

start **2** $\rightarrow$

- Parallel SFA matching



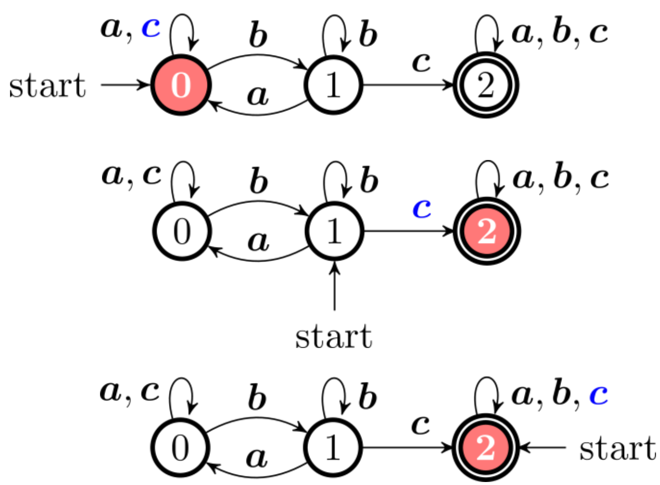
$\mathfrak{s}_0$	$\langle 0,1,2\rangle$	$\mathfrak{s}_1$	$\langle 1,1,2\rangle$	$\mathfrak{s}_2$	$\langle 0,0,2\rangle$
$\mathfrak{s}_3$	$\langle 0,2,2\rangle$	$\mathfrak{s}_4$	$\langle 2,2,2\rangle$	$\mathfrak{s}_5$	$\langle 1,2,2\rangle$

chunks: c_0 **b b a** c_1 **c a a** c_2 **b c c**

start $\mathfrak{s}_0$ $\rightarrow$

Motivation (cont.)

- Parallel FA matching



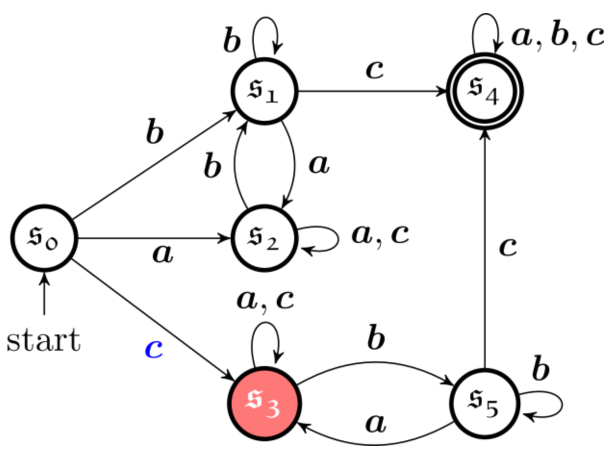
chunks: c_0 **b b a** c_1 **c a a** c_2 **b c c**

start $0 \rightarrow$ **0**

start $1 \rightarrow$ **2**

start $2 \rightarrow$ **2**

- Parallel SFA matching



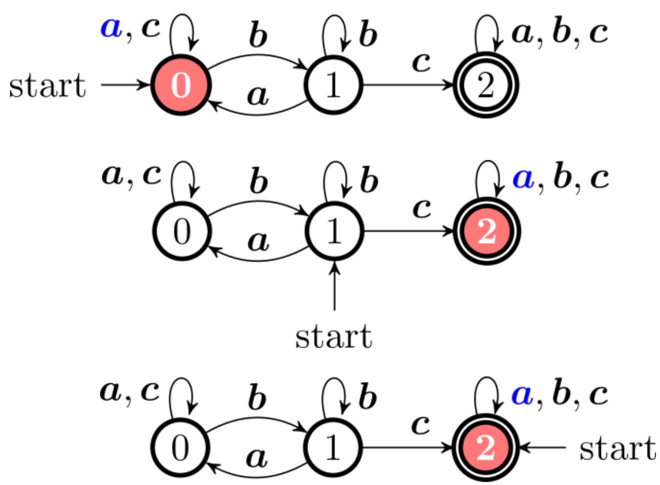
$\mathfrak{s}_0$	$\langle 0, 1, 2 \rangle$	$\mathfrak{s}_1$	$\langle 1, 1, 2 \rangle$	$\mathfrak{s}_2$	$\langle 0, 0, 2 \rangle$
$\mathfrak{s}_3$	$\langle 0, 2, 2 \rangle$	$\mathfrak{s}_4$	$\langle 2, 2, 2 \rangle$	$\mathfrak{s}_5$	$\langle 1, 2, 2 \rangle$

chunks: c_0 **b b a** c_1 **c a a** c_2 **b c c**

start $\mathfrak{s}_0 \rightarrow$ $\mathfrak{s}_3$

Motivation (cont.)

- Parallel FA matching



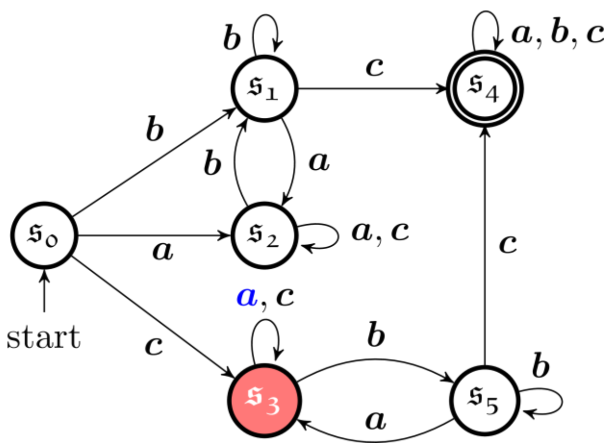
chunks: c_0 **bba** c_1 **caa** c_2 **bcc**

start 0 → 0 0

start 1 → 2 2

start 2 → 2 2

- Parallel SFA matching



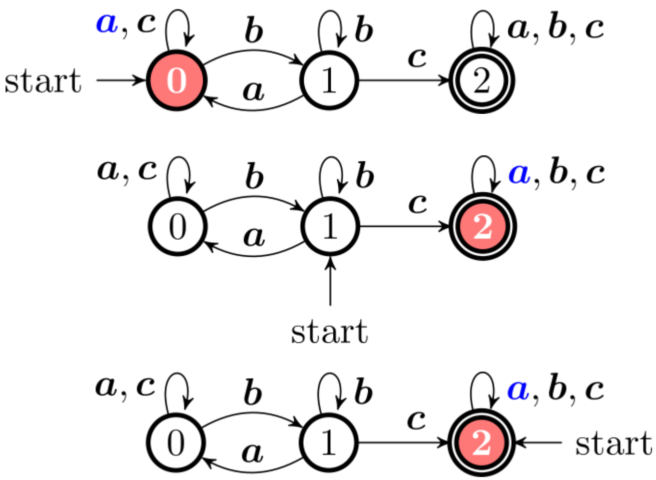
$\mathfrak{s}_0$	$\langle 0,1,2\rangle$	$\mathfrak{s}_1$	$\langle 1,1,2\rangle$	$\mathfrak{s}_2$	$\langle 0,0,2\rangle$
$\mathfrak{s}_3$	$\langle 0,2,2\rangle$	$\mathfrak{s}_4$	$\langle 2,2,2\rangle$	$\mathfrak{s}_5$	$\langle 1,2,2\rangle$

chunks: c_0 **bba** c_1 **caa** c_2 **bcc**

start $\mathfrak{s}_0$ → $\mathfrak{s}_3$ $\mathfrak{s}_3$

Motivation (cont.)

- Parallel FA matching



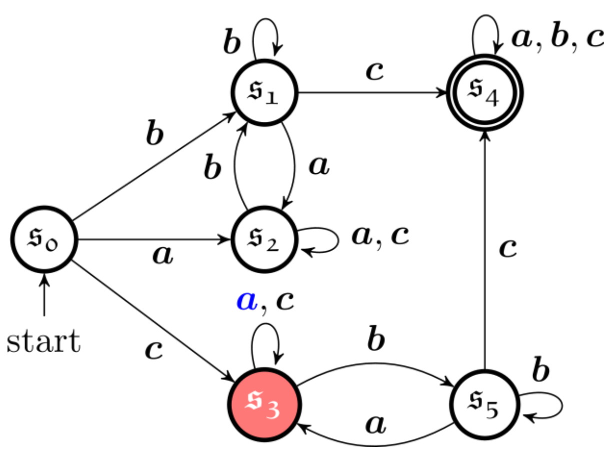
chunks: c_0 **bba** c_1 **caa** c_2 **bcc**

start 0 → 0 0 0

start 1 → 2 2 2

start 2 → 2 2 2

- Parallel SFA matching



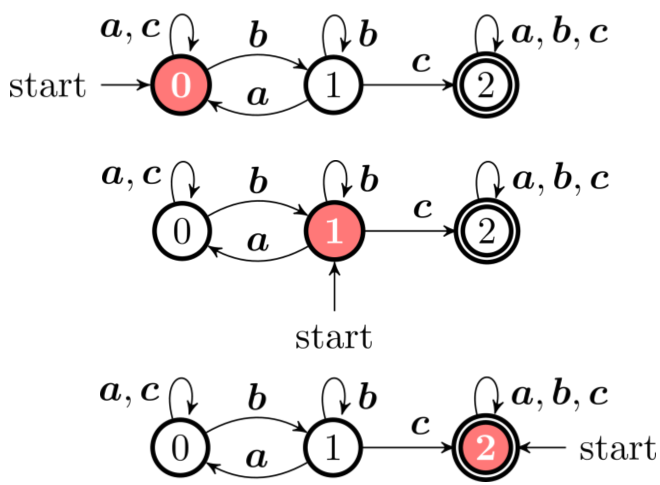
$\mathfrak{s}_0$	$\langle 0, 1, 2 \rangle$	$\mathfrak{s}_1$	$\langle 1, 1, 2 \rangle$	$\mathfrak{s}_2$	$\langle 0, 0, 2 \rangle$
$\mathfrak{s}_3$	$\langle 0, 2, 2 \rangle$	$\mathfrak{s}_4$	$\langle 2, 2, 2 \rangle$	$\mathfrak{s}_5$	$\langle 1, 2, 2 \rangle$

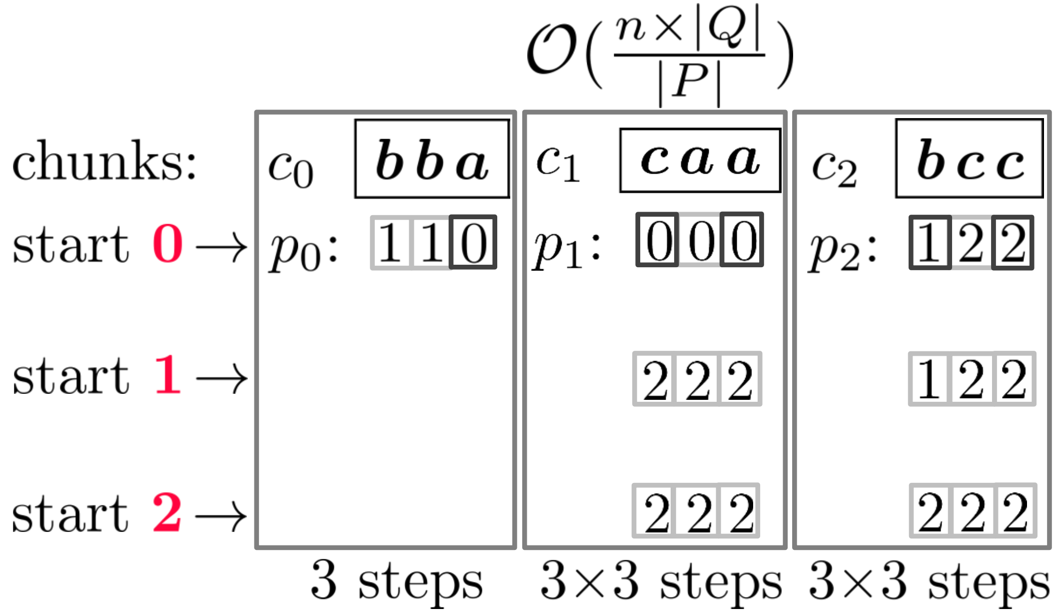
chunks: c_0 **bba** c_1 **caa** c_2 **bcc**

start $\mathfrak{s}_0$ → $\mathfrak{s}_3 \mathfrak{s}_3 \mathfrak{s}_3$

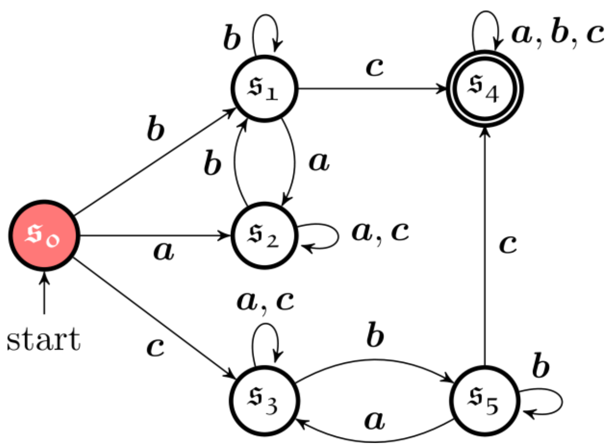
Motivation (cont.)

- Parallel FA matching



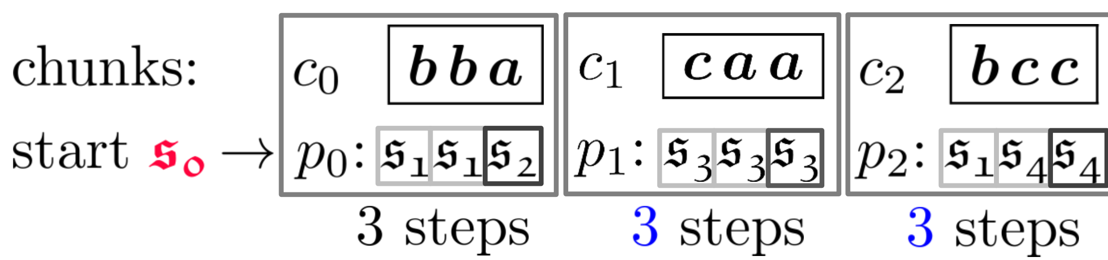


- Parallel SFA matching

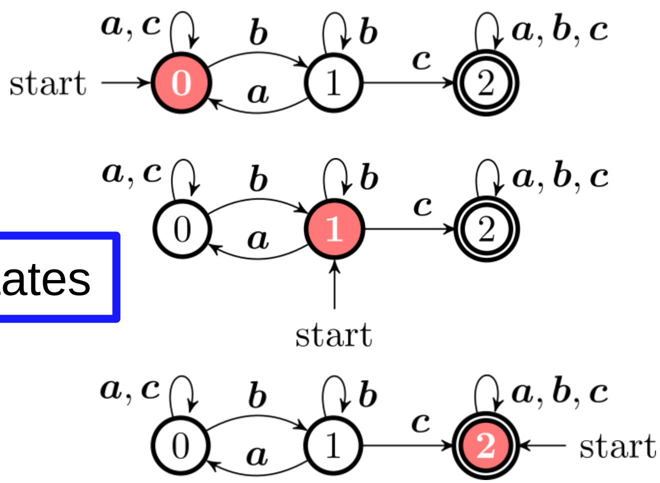


$$\mathcal{O}(\frac{n}{|P|})$$

$\mathfrak{s}_0$	$\langle 0,1,2\rangle$	$\mathfrak{s}_1$	$\langle 1,1,2\rangle$	$\mathfrak{s}_2$	$\langle 0,0,2\rangle$
$\mathfrak{s}_3$	$\langle 0,2,2\rangle$	$\mathfrak{s}_4$	$\langle 2,2,2\rangle$	$\mathfrak{s}_5$	$\langle 1,2,2\rangle$



Motivation (cont.)



chunks: c_0 **b b a** | c_1 **c a a** | c_2 **b c c**

start **0** → p_0: 1 1 0 | p_1: 0 0 0 | p_2: 1 2 2

start **1** → | 2 2 2 | 1 2 2

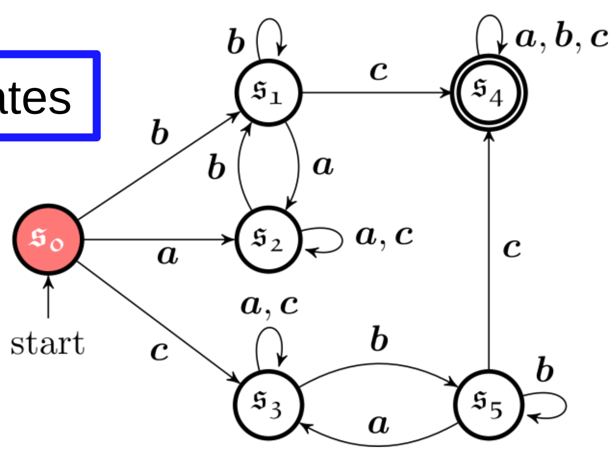
start **2** → | 2 2 2 | 2 2 2

3 steps | 3×3 steps | 3×3 steps

3 states

Problem of SFA construction:
SFA size is exponential in number of FA-states $\mathcal{O}(|Q|^{|Q|})$

6 states



$\mathfrak{s}_0$	$\langle 0,1,2\rangle$	$\mathfrak{s}_1$	$\langle 1,1,2\rangle$	$\mathfrak{s}_2$	$\langle 0,0,2\rangle$
$\mathfrak{s}_3$	$\langle 0,2,2\rangle$	$\mathfrak{s}_4$	$\langle 2,2,2\rangle$	$\mathfrak{s}_5$	$\langle 1,2,2\rangle$

chunks: c_0 **b b a** | c_1 **c a a** | c_2 **b c c**

start $\mathfrak{s}_0$ → p_0: $\mathfrak{s}_1 \mathfrak{s}_1 \mathfrak{s}_2$ | p_1: $\mathfrak{s}_3 \mathfrak{s}_3 \mathfrak{s}_3$ | p_2: $\mathfrak{s}_1 \mathfrak{s}_4 \mathfrak{s}_4$

3 steps | 3 steps | 3 steps

Our contributions

1. Introduce **fingerprint-based hashing** of SFA-states to speed up state comparisons.
2. Provide **x86 SIMD-based transposition kernels** for SFA-state construction to leverage data-parallelism and cache-locality.
3. Perform **in-memory compression** of SFA-states to mitigate the space constraints of large problems.
4. **Parallelize** SFA construction for shared-memory multicores with lock-free synchronization on all data-structures including **thread-local queues** supporting work-stealing.

Sequential SFA construction

$Q_s \leftarrow \emptyset, Q_{tmp} \leftarrow \{\mathfrak{s}_I\}$

while $Q_{tmp} \neq \emptyset$ **do**

  choose and remove a SFA state $\mathfrak{s}$ from Q_{tmp}

  $Q_s \leftarrow Q_s \cup \{\mathfrak{s}\}$

  forall the $\sigma \in \Sigma$ **do**

    $q \in Q \quad \mathfrak{s}_{next}(q) := \bigcup_{q' \in \mathfrak{s}(q)} \delta(q', \sigma)$

    $\delta_s[\mathfrak{s}, \sigma] \leftarrow \mathfrak{s}_{next}$

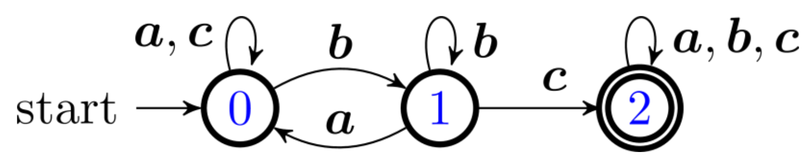
    if $\mathfrak{s}_{next} \notin Q_s, Q_{tmp}$ **then**

      $Q_{tmp} \leftarrow Q_{tmp} \cup \{\mathfrak{s}_{next}\}$

$I_s \leftarrow \{\mathfrak{s}_I\}$

$F_s \leftarrow \{\mathfrak{s} \in Q_s | \exists q \in I | \mathfrak{s}(q) \cap F \neq \emptyset\}$

Start with the initial state $\mathfrak{s}_I$.

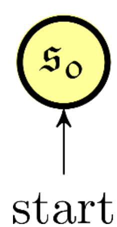


DFA over $\Sigma = \{b, a, c\}$

$Q_s = \{\}$

$Q_{tmp} = \{\mathfrak{s}_I\}$

$\mathfrak{s}_o$	$\langle 0, 1, 2 \rangle$



SFA

Sequential SFA construction

$Q_s \leftarrow \emptyset, Q_{tmp} \leftarrow \{\mathfrak{s}_I\}$

while $Q_{tmp} \neq \emptyset$ **do**

  choose and remove a SFA state $\mathfrak{s}$ from Q_{tmp}

  $Q_s \leftarrow Q_s \cup \{\mathfrak{s}\}$

  forall the $\sigma \in \Sigma$ **do**

    $q \in Q \quad \mathfrak{s}_{next}(q) := \bigcup_{q' \in \mathfrak{s}(q)} \delta(q', \sigma)$

    $\delta_s[\mathfrak{s}, \sigma] \leftarrow \mathfrak{s}_{next}$

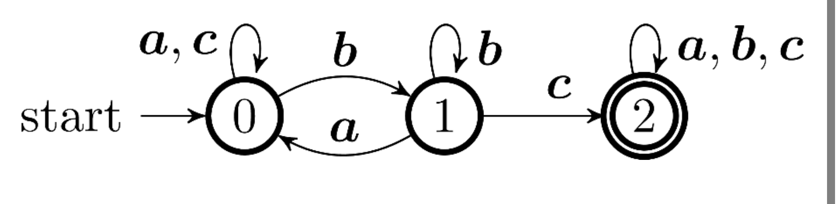
    if $\mathfrak{s}_{next} \notin Q_s, Q_{tmp}$ **then**

      $Q_{tmp} \leftarrow Q_{tmp} \cup \{\mathfrak{s}_{next}\}$

$I_s \leftarrow \{\mathfrak{s}_I\}$

$F_s \leftarrow \{\mathfrak{s} \in Q_s | \exists q \in I | \mathfrak{s}(q) \cap F \neq \emptyset\}$

Until no more states to process

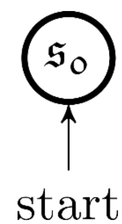


DFA over $\Sigma = \{b, a, c\}$

$Q_s = \{\}$

$Q_{tmp} = \{\mathfrak{s}_0\}$

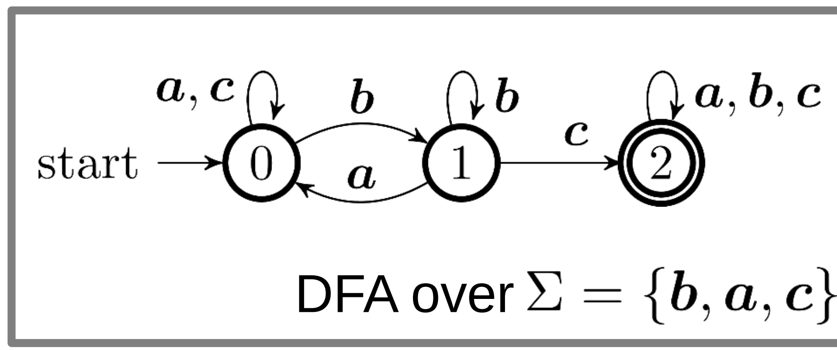
$\mathfrak{s}_0$	$\langle 0, 1, 2 \rangle$



SFA

Sequential SFA construction

$Q_s \leftarrow \emptyset, Q_{tmp} \leftarrow \{\mathfrak{s}_I\}$
while $Q_{tmp} \neq \emptyset$ **do**
  choose and remove a SFA state $\mathfrak{s}$ from Q_{tmp}
  $Q_s \leftarrow Q_s \cup \{\mathfrak{s}\}$
  forall the $\sigma \in \Sigma$ **do**
    $q \in Q \quad \mathfrak{s}_{next}(q) := \bigcup_{q' \in \mathfrak{s}(q)} \delta(q', \sigma)$
    $\delta_s[\mathfrak{s}, \sigma] \leftarrow \mathfrak{s}_{next}$
    if $\mathfrak{s}_{next} \notin Q_s, Q_{tmp}$ **then**
      $Q_{tmp} \leftarrow Q_{tmp} \cup \{\mathfrak{s}_{next}\}$
$I_s \leftarrow \{\mathfrak{s}_I\}$
$F_s \leftarrow \{\mathfrak{s} \in Q_s | \exists q \in I | \mathfrak{s}(q) \cap F \neq \emptyset\}$

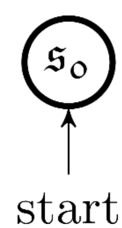


$Q_s = \{\}$

$Q_{tmp} = \{\mathfrak{s}_0\}$

$\mathfrak{s} = \mathfrak{s}_0$

$\mathfrak{s}_0$	$\langle 0, 1, 2 \rangle$

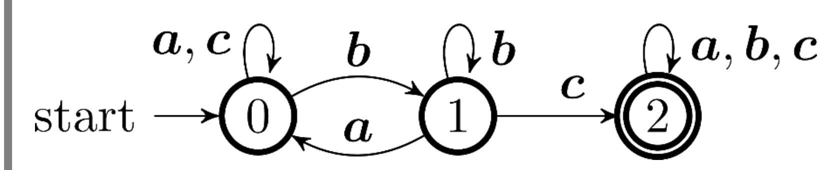


SFA

Sequential SFA construction

$Q_\text{s} \leftarrow \emptyset, Q_\text{tmp} \leftarrow \{\mathfrak{s}_\text{I}\}$
while $Q_\text{tmp} \neq \emptyset$ **do**
  choose and remove a SFA state $\mathfrak{s}$ from Q_tmp
  $Q_\text{s} \leftarrow Q_\text{s} \cup \{\mathfrak{s}\}$
  forall the $\sigma \in \Sigma$ **do**
    $q \in Q \quad \mathfrak{s}_\text{next}(q) := \bigcup_{q' \in \mathfrak{s}(q)} \delta(q', \sigma)$
    $\delta_\text{s}[\mathfrak{s}, \sigma] \leftarrow \mathfrak{s}_\text{next}$
    if $\mathfrak{s}_\text{next} \notin Q_\text{s}, Q_\text{tmp}$ **then**
      $Q_\text{tmp} \leftarrow Q_\text{tmp} \cup \{\mathfrak{s}_\text{next}\}$
$I_\text{s} \leftarrow \{\mathfrak{s}_\text{I}\}$
$F_\text{s} \leftarrow \{\mathfrak{s} \in Q_\text{s} | \exists q \in I | \mathfrak{s}(q) \cap F \neq \emptyset\}$

Insert $\mathfrak{s}$ into the processed set

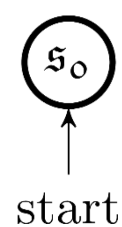


DFA over $\Sigma = \{b, a, c\}$

$Q_\text{s} = \{\mathfrak{s}_0\}$
$Q_\text{tmp} = \{\}$

$\mathfrak{s} = \mathfrak{s}_0$

$\mathfrak{s}_0$	$\langle 0, 1, 2 \rangle$

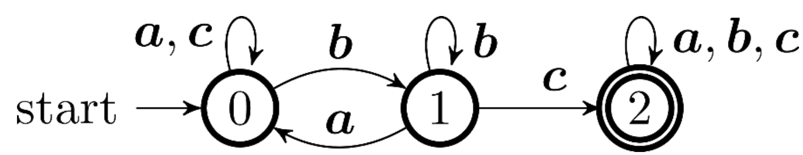


SFA

Sequential SFA construction

$Q_s \leftarrow \emptyset, Q_{tmp} \leftarrow \{\mathfrak{s}_I\}$
while $Q_{tmp} \neq \emptyset$ **do**
  choose and remove a SFA state $\mathfrak{s}$ from Q_{tmp}
  $Q_s \leftarrow Q_s \cup \{\mathfrak{s}\}$
  forall the $\sigma \in \Sigma$ **do**
    $q \in Q \quad \mathfrak{s}_{next}(q) := \bigcup_{q' \in \mathfrak{s}(q)} \delta(q', \sigma)$
    $\delta_s[\mathfrak{s}, \sigma] \leftarrow \mathfrak{s}_{next}$
    if $\mathfrak{s}_{next} \notin Q_s, Q_{tmp}$ **then**
      $Q_{tmp} \leftarrow Q_{tmp} \cup \{\mathfrak{s}_{next}\}$
$I_s \leftarrow \{\mathfrak{s}_I\}$
$F_s \leftarrow \{\mathfrak{s} \in Q_s | \exists q \in I | \mathfrak{s}(q) \cap F \neq \emptyset\}$

Iterate with every symbol

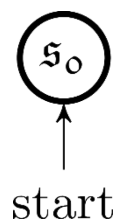


DFA over $\Sigma = \{b, a, c\}$

$$Q_s = \{\mathfrak{s}_o\}$$
$$Q_{tmp} = \{\}$$

$$\mathfrak{s} = \mathfrak{s}_o$$

$\mathfrak{s}_o$	$\langle 0, 1, 2 \rangle$

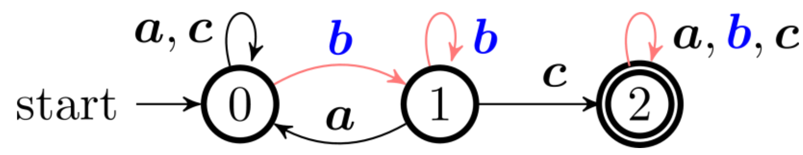


SFA

Sequential SFA construction

$Q_s \leftarrow \emptyset, Q_{tmp} \leftarrow \{\mathfrak{s}_I\}$
while $Q_{tmp} \neq \emptyset$ **do**
  choose and remove a SFA state $\mathfrak{s}$ from Q_{tmp}
  $Q_s \leftarrow Q_s \cup \{\mathfrak{s}\}$
  forall the $\sigma \in \Sigma$ **do**
    $q \in Q \quad \mathfrak{s}_{next}(q) := \bigcup_{q' \in \mathfrak{s}(q)} \delta(q', \sigma)$
    $\delta_s[\mathfrak{s}, \sigma] \leftarrow \mathfrak{s}_{next}$
    if $\mathfrak{s}_{next} \notin Q_s, Q_{tmp}$ **then**
      $Q_{tmp} \leftarrow Q_{tmp} \cup \{\mathfrak{s}_{next}\}$
$I_s \leftarrow \{\mathfrak{s}_I\}$
$F_s \leftarrow \{\mathfrak{s} \in Q_s | \exists q \in I | \mathfrak{s}(q) \cap F \neq \emptyset\}$

Find new states



DFA over $\Sigma = \{b, a, c\}$

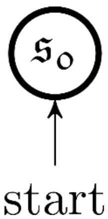
$Q_s = \{\mathfrak{s}_0\}$
$Q_{tmp} = \{\}$

$\mathfrak{s}_0$	$\langle 0, 1, 2 \rangle$

$\mathfrak{s}_0 = \{0, 1, 2\}$

$\downarrow \downarrow \downarrow \; b$

$\mathfrak{s}_{next} = \{1, 1, 2\}$



SFA

Sequential SFA construction

$Q_s \leftarrow \emptyset, Q_{tmp} \leftarrow \{\mathfrak{s}_I\}$

while $Q_{tmp} \neq \emptyset$ **do**

  choose and remove a SFA state $\mathfrak{s}$ from Q_{tmp}

  $Q_s \leftarrow Q_s \cup \{\mathfrak{s}\}$

  forall the $\sigma \in \Sigma$ **do**

    $q \in Q \quad \mathfrak{s}_{next}(q) := \bigcup_{q' \in \mathfrak{s}(q)} \delta(q', \sigma)$

    $\delta_s[\mathfrak{s}, \sigma] \leftarrow \mathfrak{s}_{next}$

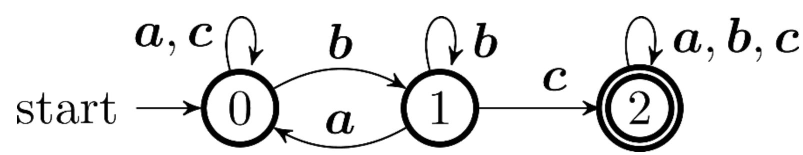
    if $\mathfrak{s}_{next} \notin Q_s, Q_{tmp}$ **then**

      $Q_{tmp} \leftarrow Q_{tmp} \cup \{\mathfrak{s}_{next}\}$

$I_s \leftarrow \{\mathfrak{s}_I\}$

$F_s \leftarrow \{\mathfrak{s} \in Q_s | \exists q \in I | \mathfrak{s}(q) \cap F \neq \emptyset\}$

Update the SFA transition function



DFA over $\Sigma = \{b, a, c\}$

$$Q_s = \{\mathfrak{s}_0\}$$

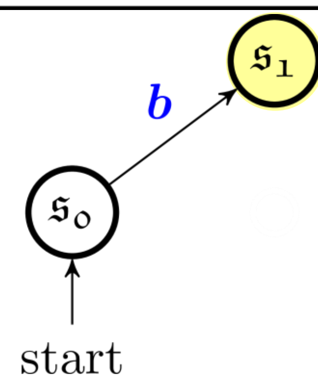
$$Q_{tmp} = \{\}$$

$$\mathfrak{s}_0 = \{0, 1, 2\}$$

$$\downarrow \downarrow \downarrow b$$

$$\mathfrak{s}_1 = \{1, 1, 2\}$$

$\mathfrak{s}_0$	$\langle 0, 1, 2 \rangle$
$\mathfrak{s}_1$	$\langle 1, 1, 2 \rangle$

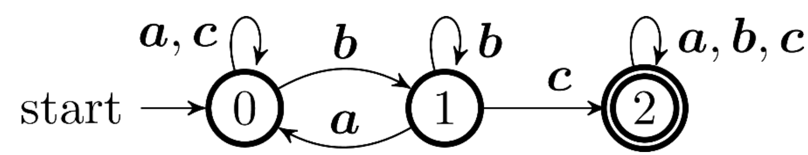


SFA

Sequential SFA construction

$Q_s \leftarrow \emptyset, Q_{tmp} \leftarrow \{\mathfrak{s}_I\}$
while $Q_{tmp} \neq \emptyset$ **do**
  choose and remove a SFA state $\mathfrak{s}$ from Q_{tmp}
  $Q_s \leftarrow Q_s \cup \{\mathfrak{s}\}$
  forall the $\sigma \in \Sigma$ **do**
    $q \in Q \quad \mathfrak{s}_{next}(q) := \bigcup_{q' \in \mathfrak{s}(q)} \delta(q', \sigma)$
    $\delta_s[\mathfrak{s}, \sigma] \leftarrow \mathfrak{s}_{next}$
    if $\mathfrak{s}_{next} \notin Q_s, Q_{tmp}$ **then**
      $Q_{tmp} \leftarrow Q_{tmp} \cup \{\mathfrak{s}_{next}\}$
$I_s \leftarrow \{\mathfrak{s}_I\}$
$F_s \leftarrow \{\mathfrak{s} \in Q_s | \exists q \in I | \mathfrak{s}(q) \cap F \neq \emptyset\}$

Check existence & add new state to the set (set membership test)



DFA over $\Sigma = \{b, a, c\}$

$Q_s = \{\mathfrak{s}_0\}$

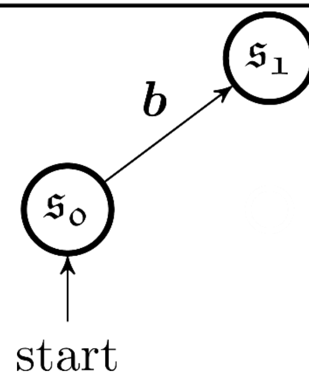
$Q_{tmp} = \{\mathfrak{s}_1\}$

$\mathfrak{s}_0 = \{0, 1, 2\}$

$\downarrow \downarrow \downarrow b$

$\mathfrak{s}_1 = \{1, 1, 2\}$

$\mathfrak{s}_0$	$\langle 0, 1, 2 \rangle$
$\mathfrak{s}_1$	$\langle 1, 1, 2 \rangle$

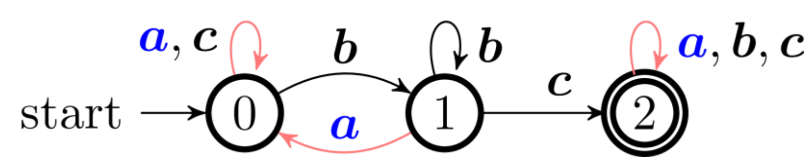


SFA

Sequential SFA construction

$Q_s \leftarrow \emptyset, Q_{tmp} \leftarrow \{\mathfrak{s}_I\}$
while $Q_{tmp} \neq \emptyset$ **do**
  choose and remove a SFA state $\mathfrak{s}$ from Q_{tmp}
  $Q_s \leftarrow Q_s \cup \{\mathfrak{s}\}$
  forall the $\sigma \in \Sigma$ **do**
    $q \in Q \quad \mathfrak{s}_{next}(q) := \bigcup_{q' \in \mathfrak{s}(q)} \delta(q', \sigma)$
    $\delta_s[\mathfrak{s}, \sigma] \leftarrow \mathfrak{s}_{next}$
    if $\mathfrak{s}_{next} \notin Q_s, Q_{tmp}$ **then**
      $Q_{tmp} \leftarrow Q_{tmp} \cup \{\mathfrak{s}_{next}\}$
$I_s \leftarrow \{\mathfrak{s}_I\}$
$F_s \leftarrow \{\mathfrak{s} \in Q_s | \exists q \in I | \mathfrak{s}(q) \cap F \neq \emptyset\}$

Generate a next state with symbol a



DFA over $\Sigma = \{b, a, c\}$

$Q_s = \{\mathfrak{s}_0\}$

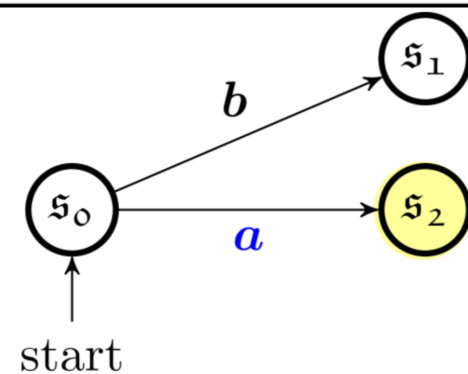
$Q_{tmp} = \{\mathfrak{s}_1, \mathfrak{s}_2\}$

$\mathfrak{s}_0 = \{0, 1, 2\}$

$\downarrow \downarrow \downarrow a$

$\mathfrak{s}_2 = \{0, 0, 2\}$

$\mathfrak{s}_0$	$\langle 0, 1, 2 \rangle$
$\mathfrak{s}_1$	$\langle 1, 1, 2 \rangle$
$\mathfrak{s}_2$	$\langle 0, 0, 2 \rangle$

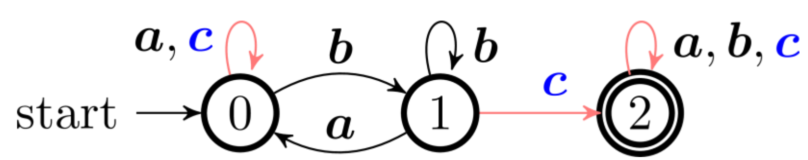


SFA

Sequential SFA construction

$Q_{\mathrm{s}} \leftarrow \emptyset, Q_{\mathrm{tmp}} \leftarrow \{\mathfrak{s}_{\mathrm{I}}\}$
while $Q_{\mathrm{tmp}} \neq \emptyset$ **do**
    choose and remove a SFA state $\mathfrak{s}$ from Q_{tmp}
    $Q_{\mathrm{s}} \leftarrow Q_{\mathrm{s}} \cup \{\mathfrak{s}\}$
    forall the $\sigma \in \Sigma$ **do**
        $q \in Q \quad \mathfrak{s}_{\mathrm{next}}(q) := \bigcup_{q' \in \mathfrak{s}(q)} \delta(q', \sigma)$
        $\delta_{\mathrm{s}}[\mathfrak{s}, \sigma] \leftarrow \mathfrak{s}_{\mathrm{next}}$
        if $\mathfrak{s}_{\mathrm{next}} \notin Q_{\mathrm{s}}, Q_{\mathrm{tmp}}$ **then**
            $Q_{\mathrm{tmp}} \leftarrow Q_{\mathrm{tmp}} \cup \{\mathfrak{s}_{\mathrm{next}}\}$
$I_{\mathrm{s}} \leftarrow \{\mathfrak{s}_{\mathrm{I}}\}$
$F_{\mathrm{s}} \leftarrow \{\mathfrak{s} \in Q_{\mathrm{s}} | \exists q \in I | \mathfrak{s}(q) \cap F \neq \emptyset\}$

Generate a next state with symbol c



DFA over $\Sigma = \{b, a, c\}$

$Q_{\mathrm{s}} = \{\mathfrak{s}_0\}$

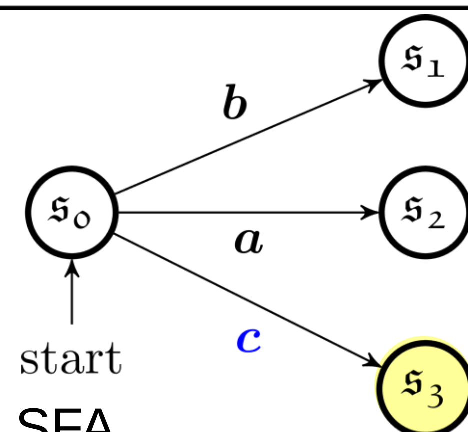
$Q_{\mathrm{tmp}} = \{\mathfrak{s}_1, \mathfrak{s}_2, \mathfrak{s}_3\}$

$\mathfrak{s}_0 = \{0, 1, 2\}$

$\downarrow \downarrow \downarrow$ c

$\mathfrak{s}_3 = \{0, 2, 2\}$

$\mathfrak{s}_0$	$\langle 0, 1, 2 \rangle$
$\mathfrak{s}_1$	$\langle 1, 1, 2 \rangle$
$\mathfrak{s}_2$	$\langle 0, 0, 2 \rangle$
$\mathfrak{s}_3$	$\langle 0, 2, 2 \rangle$

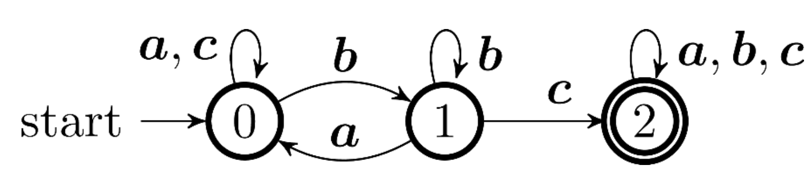


SFA

Sequential SFA construction

$Q_s \leftarrow \emptyset, Q_{tmp} \leftarrow \{\mathfrak{s}_I\}$
while $Q_{tmp} \neq \emptyset$ **do**
  choose and remove a SFA state $\mathfrak{s}$ from Q_{tmp}
  $Q_s \leftarrow Q_s \cup \{\mathfrak{s}\}$
  forall the $\sigma \in \Sigma$ **do**
    $q \in Q \quad \mathfrak{s}_{next}(q) := \bigcup_{q' \in \mathfrak{s}(q)} \delta(q', \sigma)$
    $\delta_s[\mathfrak{s}, \sigma] \leftarrow \mathfrak{s}_{next}$
    if $\mathfrak{s}_{next} \notin Q_s, Q_{tmp}$ **then**
      $Q_{tmp} \leftarrow Q_{tmp} \cup \{\mathfrak{s}_{next}\}$
$I_s \leftarrow \{\mathfrak{s}_I\}$
$F_s \leftarrow \{\mathfrak{s} \in Q_s | \exists q \in I | \mathfrak{s}(q) \cap F \neq \emptyset\}$

Choose the unprocessed state $\mathfrak{s}_1$

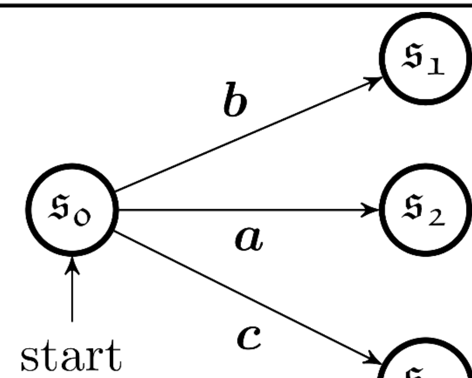


DFA over $\Sigma = \{b, a, c\}$

$Q_s = \{\mathfrak{s}_0, \mathfrak{s}_1\}$
$Q_{tmp} = \{\mathfrak{s}_2, \mathfrak{s}_3\}$

$\mathfrak{s} = \mathfrak{s}_1$

$\mathfrak{s}_0$	$\langle 0, 1, 2 \rangle$
$\mathfrak{s}_1$	$\langle 1, 1, 2 \rangle$
$\mathfrak{s}_2$	$\langle 0, 0, 2 \rangle$
$\mathfrak{s}_3$	$\langle 0, 2, 2 \rangle$

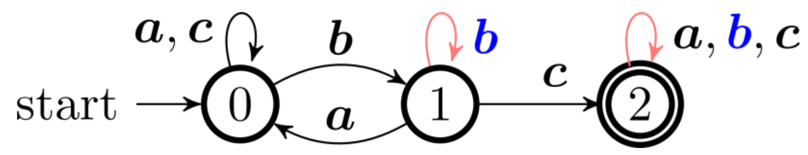


SFA

Sequential SFA construction

$Q_s \leftarrow \emptyset, Q_{tmp} \leftarrow \{\mathfrak{s}_I\}$
while $Q_{tmp} \neq \emptyset$ **do**
  choose and remove a SFA state $\mathfrak{s}$ from Q_{tmp}
  $Q_s \leftarrow Q_s \cup \{\mathfrak{s}\}$
  forall the $\sigma \in \Sigma$ **do**
    $q \in Q \quad \mathfrak{s}_{next}(q) := \bigcup_{q' \in \mathfrak{s}(q)} \delta(q', \sigma)$
    $\delta_s[\mathfrak{s}, \sigma] \leftarrow \mathfrak{s}_{next}$
    if $\mathfrak{s}_{next} \notin Q_s, Q_{tmp}$ **then**
      $Q_{tmp} \leftarrow Q_{tmp} \cup \{\mathfrak{s}_{next}\}$
$I_s \leftarrow \{\mathfrak{s}_I\}$
$F_s \leftarrow \{\mathfrak{s} \in Q_s | \exists q \in I | \mathfrak{s}(q) \cap F \neq \emptyset\}$

Generate a next state with symbol ***b***

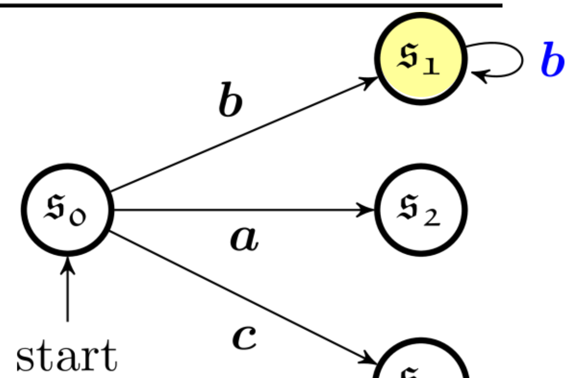


DFA over $\Sigma = \{b, a, c\}$

$Q_s = \{\mathfrak{s}_0, \mathfrak{s}_1\}$
$Q_{tmp} = \{\mathfrak{s}_2, \mathfrak{s}_3\}$

$\mathfrak{s}_1 = \{1, 1, 2\}$
$\downarrow\downarrow\downarrow$ ***b***
$\mathfrak{s}_{next} = \{1, 1, 2\}$

$\mathfrak{s}_0$	$\langle 0, 1, 2 \rangle$
$\mathfrak{s}_1$	$\langle 1, 1, 2 \rangle$
$\mathfrak{s}_2$	$\langle 0, 0, 2 \rangle$
$\mathfrak{s}_3$	$\langle 0, 2, 2 \rangle$



SFA

Sequential SFA construction

$Q_\mathrm{s} \leftarrow \emptyset, Q_\mathrm{tmp} \leftarrow \{\mathfrak{s}_\mathrm{I}\}$

while $Q_\mathrm{tmp} \neq \emptyset$ **do**

  choose and remove a SFA state $\mathfrak{s}$ from Q_tmp

  $Q_\mathrm{s} \leftarrow Q_\mathrm{s} \cup \{\mathfrak{s}\}$

  forall the $\sigma \in \Sigma$ **do**

    $q \in Q \quad \mathfrak{s}_\mathrm{next}(q) := \bigcup_{q' \in \mathfrak{s}(q)} \delta(q', \sigma)$

    $\delta_\mathrm{s}[\mathfrak{s}, \sigma] \leftarrow \mathfrak{s}_\mathrm{next}$

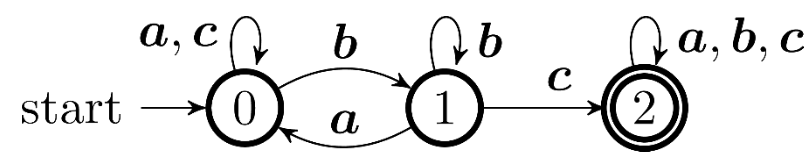
    if $\mathfrak{s}_\mathrm{next} \notin Q_\mathrm{s}, Q_\mathrm{tmp}$ **then**

      $Q_\mathrm{tmp} \leftarrow Q_\mathrm{tmp} \cup \{\mathfrak{s}_\mathrm{next}\}$

$I_\mathrm{s} \leftarrow \{\mathfrak{s}_\mathrm{I}\}$

$F_\mathrm{s} \leftarrow \{\mathfrak{s} \in Q_\mathrm{s} | \exists q \in I | \mathfrak{s}(q) \cap F \neq \emptyset\}$

Until no more states to process

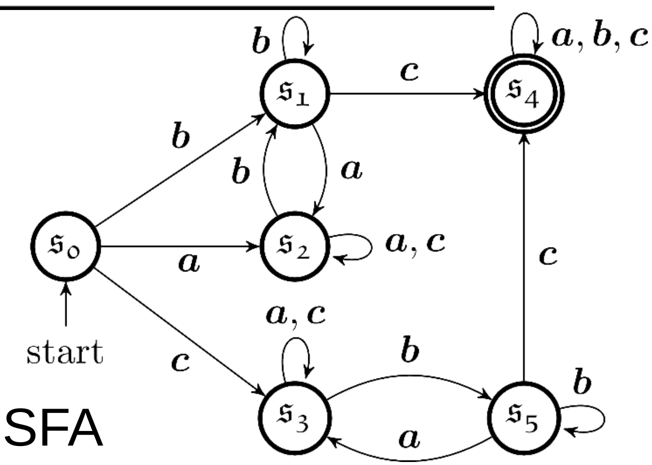


DFA over $\Sigma = \{b, a, c\}$

$Q_\mathrm{s} = \{\mathfrak{s}_0, \mathfrak{s}_1, \mathfrak{s}_2, \mathfrak{s}_3, \mathfrak{s}_4, \mathfrak{s}_5, \mathfrak{s}_6\}$

$Q_\mathrm{tmp} = \{\}$

$\mathfrak{s}_0$	$\langle 0,1,2 \rangle$	$\mathfrak{s}_1$	$\langle 1,1,2 \rangle$	$\mathfrak{s}_2$	$\langle 0,0,2 \rangle$
$\mathfrak{s}_3$	$\langle 0,2,2 \rangle$	$\mathfrak{s}_4$	$\langle 2,2,2 \rangle$	$\mathfrak{s}_5$	$\langle 1,2,2 \rangle$

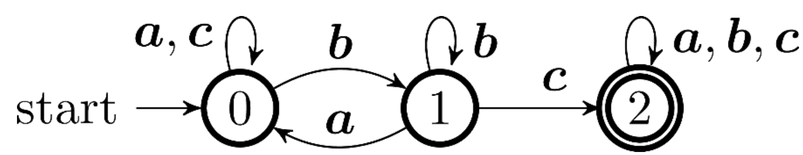


SFA

Sequential SFA construction

$Q_s \leftarrow \emptyset, Q_{tmp} \leftarrow \{\mathfrak{s}_I\}$
while $Q_{tmp} \neq \emptyset$ **do**
  choose and remove a SFA state $\mathfrak{s}$ from Q_{tmp}
  $Q_s \leftarrow Q_s \cup \{\mathfrak{s}\}$
  forall the $\sigma \in \Sigma$ **do**
    $q \in Q \quad \mathfrak{s}_{next}(q) := \bigcup_{q' \in \mathfrak{s}(q)} \delta(q', \sigma)$
    $\delta_s[\mathfrak{s}, \sigma] \leftarrow \mathfrak{s}_{next}$
    if $\mathfrak{s}_{next} \notin Q_s, Q_{tmp}$ **then**
      $Q_{tmp} \leftarrow Q_{tmp} \cup \{\mathfrak{s}_{next}\}$
$I_s \leftarrow \{\mathfrak{s}_I\}$
$F_s \leftarrow \{\mathfrak{s} \in Q_s | \exists q \in I | \mathfrak{s}(q) \cap F \neq \emptyset\}$

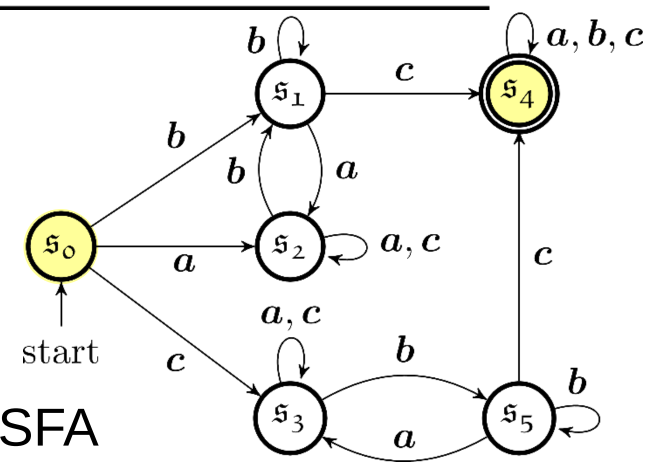
Set the initial and the final state



DFA over $\Sigma = \{b, a, c\}$

$$Q_s = \{\mathfrak{s}_0, \mathfrak{s}_1, \mathfrak{s}_2, \mathfrak{s}_3, \mathfrak{s}_4, \mathfrak{s}_5, \mathfrak{s}_6\}$$
$$Q_{tmp} = \{\}$$

$\mathfrak{s}_0$	$\langle 0, 1, 2 \rangle$	$\mathfrak{s}_1$	$\langle 1, 1, 2 \rangle$	$\mathfrak{s}_2$	$\langle 0, 0, 2 \rangle$
$\mathfrak{s}_3$	$\langle 0, 2, 2 \rangle$	$\mathfrak{s}_4$	$\langle 2, 2, 2 \rangle$	$\mathfrak{s}_5$	$\langle 1, 2, 2 \rangle$



SFA

Optimizing SFA construction

$Q_{\mathrm{s}} \leftarrow \emptyset, Q_{\mathrm{tmp}} \leftarrow \{\mathfrak{s}_{\mathrm{I}}\}$

while $Q_{\mathrm{tmp}} \neq \emptyset$ **do**

  choose and remove a SFA state $\mathfrak{s}$ from Q_{tmp}

  $Q_{\mathrm{s}} \leftarrow Q_{\mathrm{s}} \cup \{\mathfrak{s}\}$

  forall the $\sigma \in \Sigma$ **do**

    $q \in Q \quad \mathfrak{s}_{\mathrm{next}}(q) := \bigcup_{q' \in \mathfrak{s}(q)} \delta(q', \sigma)$

    $\delta_{\mathrm{s}}[\mathfrak{s}, \sigma] \leftarrow \mathfrak{s}_{\mathrm{next}}$

    if $\mathfrak{s}_{\mathrm{next}} \notin Q_{\mathrm{s}}, Q_{\mathrm{tmp}}$ **then**

      $Q_{\mathrm{tmp}} \leftarrow Q_{\mathrm{tmp}} \cup \{\mathfrak{s}_{\mathrm{next}}\}$

$I_{\mathrm{s}} \leftarrow \{\mathfrak{s}_{\mathrm{I}}\}$

$F_{\mathrm{s}} \leftarrow \{\mathfrak{s} \in Q_{\mathrm{s}} | \exists q \in I | \mathfrak{s}(q) \cap F \neq \emptyset\}$

$$\mathcal{O}\left(\sum_{i=1}^{|Q_s|} \sum_{j=1}^{|\Sigma|} (|Q| + |Q| \times i)\right) = \mathcal{O}\left(\frac{1}{2} \times |\Sigma| \times |Q| \times |Q_s| \times (|Q_s| + 3)\right)$$

An SFA size $|Q_s|$ is $\mathcal{O}(|Q|^{|Q|})$ in the worst case

Exponential state-growth

Optimizing SFA construction

$Q_{\mathrm{s}} \leftarrow \emptyset, Q_{\mathrm{tmp}} \leftarrow \{\mathfrak{s}_{\mathrm{I}}\}$

while $Q_{\mathrm{tmp}} \neq \emptyset$ **do**

  choose and remove a SFA state $\mathfrak{s}$ from Q_{tmp}

  $Q_{\mathrm{s}} \leftarrow Q_{\mathrm{s}} \cup \{\mathfrak{s}\}$

  forall the $\sigma \in \Sigma$ **do**

    $q \in Q \quad \mathfrak{s}_{\mathrm{next}}(q) := \bigcup_{q' \in \mathfrak{s}(q)} \delta(q', \sigma)$ ← Parameterized transposition

    $\delta_{\mathrm{s}}[\mathfrak{s}, \sigma] \leftarrow \mathfrak{s}_{\mathrm{next}}$

    if $\mathfrak{s}_{\mathrm{next}} \notin Q_{\mathrm{s}}, Q_{\mathrm{tmp}}$ **then** ← Fingerprint-based hashing

      $Q_{\mathrm{tmp}} \leftarrow Q_{\mathrm{tmp}} \cup \{\mathfrak{s}_{\mathrm{next}}\}$

$I_{\mathrm{s}} \leftarrow \{\mathfrak{s}_{\mathrm{I}}\}$

$F_{\mathrm{s}} \leftarrow \{\mathfrak{s} \in Q_{\mathrm{s}} | \exists q \in I | \mathfrak{s}(q) \cap F \neq \emptyset\}$

$$\mathcal{O}\left(\sum_{i=1}^{|Q_s|} \sum_{j=1}^{|\Sigma|} (|Q| + |Q| \times i)\right) = \mathcal{O}\left(\frac{1}{2} \times |\Sigma| \times |Q| \times |Q_s| \times (|Q_s| + 3)\right)$$

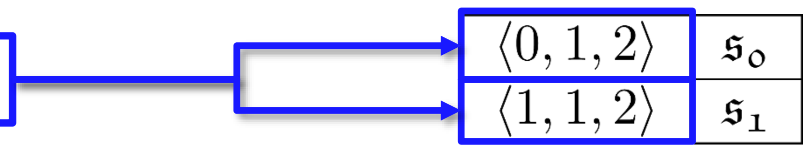
An SFA size $|Q_s|$ is $\mathcal{O}(|Q|^{|Q|})$ in the worst case

Exponential state-growth

Fingerprint-based hashing

- Fingerprints (F)
  - Short bit-strings for larger objects (SFA-states)
  - **CityHash**, FarmHash, Rabin’s method, etc. create fingerprints
  - Speed up comparisons of SFA-states

$$\mathfrak{s}_0 = \langle 0, 1, 2 \rangle \xrightarrow{a} \mathfrak{s}_{\text{next}} = \boxed{\langle 0, 0, 2 \rangle}$$



$\mathcal{O}(|Q|)$ exhaustive SFA-state comparisons

Fingerprint-based hashing

- Fingerprints (F)
  - Short bit-strings for larger objects (SFA-states)
  - **CityHash**, FarmHash, Rabin's method, etc. create fingerprints
  - Speed up comparisons of SFA-states

$$\mathfrak{s}_0 = \langle 0, 1, 2 \rangle \xrightarrow{a} \mathfrak{s}_{\text{next}} = \langle 0, 0, 2 \rangle \; \boxed{F_{\text{next}}}$$

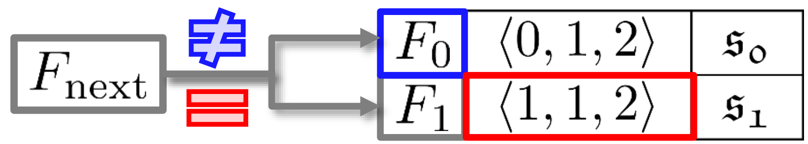
F_0	$\langle 0, 1, 2 \rangle$	$\mathfrak{s}_0$
F_1	$\langle 1, 1, 2 \rangle$	$\mathfrak{s}_1$

$\mathcal{O}(1)$ fingerprint comparisons

Fingerprint-based hashing

- Fingerprints (F)
  - Short bit-strings for larger objects (SFA-states)
  - **CityHash**, FarmHash, Rabin's method, etc. create fingerprints
  - Speed up comparisons of SFA-states

$\mathfrak{s}_0 = \langle 0, 1, 2 \rangle \xrightarrow{a} \mathfrak{s}_{\text{next}} = \langle 0, 0, 2 \rangle$



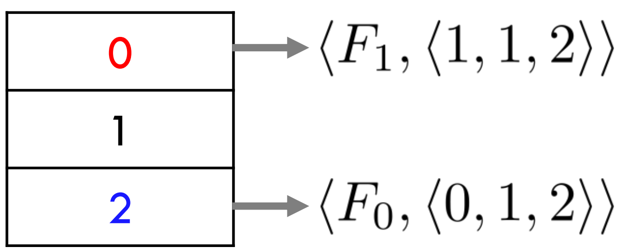
- Fingerprint-collisions
  - It follows from the properties of the hash function that if fingerprints are different, SFA-states are different.
    - No exhaustive comparison necessary.
  - With small probability, different SFA-states generate **same** fingerprint.
    - Fingerprint-collision
    - If fingerprints are the same, SFA-states **may** be the same.

→ $\mathcal{O}(|Q|)$ exhaustive comparisons are required.

Fingerprint-based hashing (cont.)

- Hashing of SFA-states
  - Speed up lookups, reduces number of SFA-state comparisons
  - **Hash key:** fingerprint % size of the hash-table
  - **Value:** $\langle$ fingerprint, SFA-state $\rangle$

$\mathfrak{s}_0$	$F_0 = 2$	$\langle 0, 1, 2 \rangle$
$\mathfrak{s}_1$	$F_1 = 0$	$\langle 1, 1, 2 \rangle$



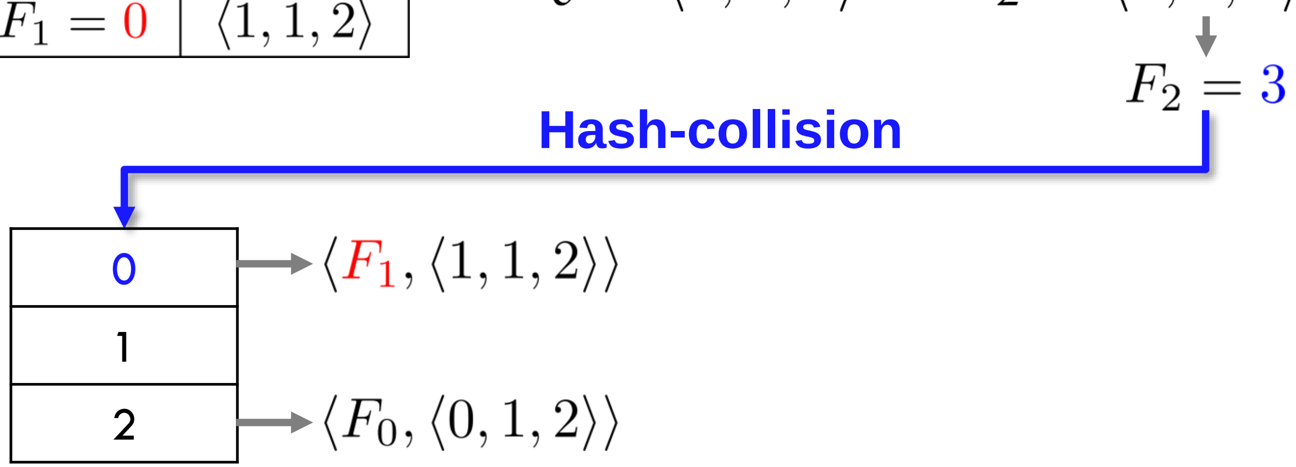
Hash-table (size=3)

Fingerprint-based hashing (cont.)

- Hash-collisions
  - Different SFA-states may map to the same hash-key due to the modulo-operation.

$\mathfrak{s}_0$	$F_0 = 2$	$\langle 0, 1, 2 \rangle$
$\mathfrak{s}_1$	$F_1 = 0$	$\langle 1, 1, 2 \rangle$

$$\mathfrak{s}_0 = \langle 0, 1, 2 \rangle \xrightarrow{a} \mathfrak{s}_2 = \langle 0, 0, 2 \rangle$$



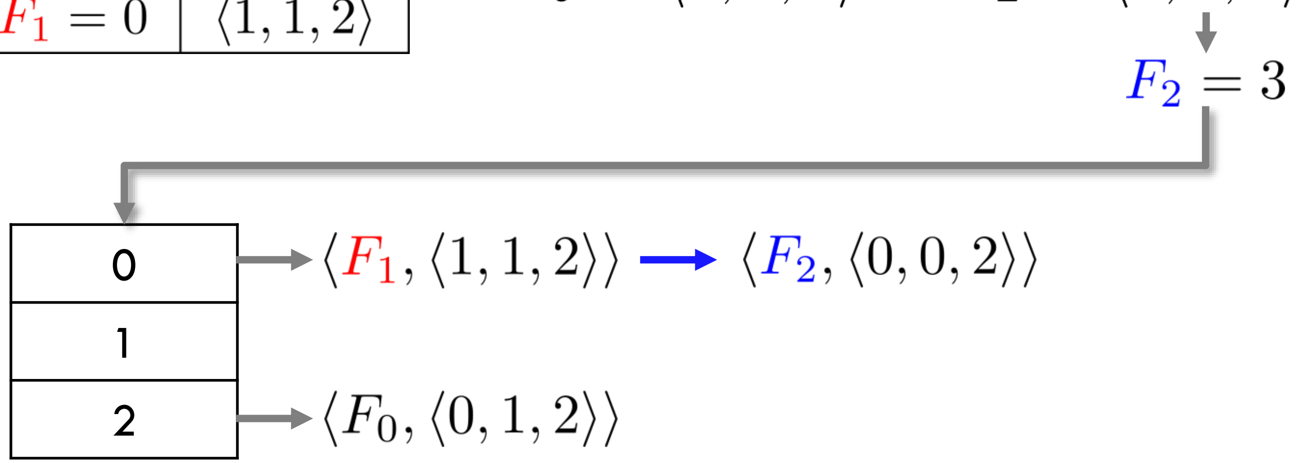
Hash-table (size=3)

Fingerprint-based hashing (cont.)

- Hash-collisions
  - Different SFA-states may map to the same hash-key due to the modulo-operation.
  - Resolved by closed addressing with chaining

$\mathfrak{s}_0$	$F_0 = 2$	$\langle 0, 1, 2 \rangle$
$\mathfrak{s}_1$	$F_1 = 0$	$\langle 1, 1, 2 \rangle$

$$\mathfrak{s}_0 = \langle 0, 1, 2 \rangle \xrightarrow{a} \mathfrak{s}_2 = \langle 0, 0, 2 \rangle$$



Hash-table (size=3)

Parameterized transposition

- Speed up creating next SFA-states of each SFA-state

$$\mathfrak{s}_0 = \langle 0, 1, 2 \rangle \xrightarrow{a} \mathfrak{s}_1 = \langle 1, 1, 2 \rangle$$
$$\xrightarrow{b} \mathfrak{s}_2 = \langle 0, 0, 2 \rangle$$
$$\xrightarrow{c} \mathfrak{s}_3 = \langle 0, 2, 2 \rangle$$

	a	b	c
0	1	0	0
1	1	0	2
2	2	2	2

DFA transition table

Non-optimized:
compute next states one by one

Parameterized transposition

- Speed up creating next SFA-states of each SFA-state

x \ y	a	b	c
0	1	0	0
1	1	0	2
2	2	2	2

DFA transition table

$$\mathfrak{s}_0 = \langle 0, 2, 0 \rangle \xrightarrow{a} \mathfrak{s}_1 = \langle 1, 2, 1 \rangle$$
$$\xrightarrow{b} \mathfrak{s}_2 = \langle 0, 2, 0 \rangle$$
$$\xrightarrow{c} \mathfrak{s}_3 = \langle 0, 2, 0 \rangle$$

$$\mathfrak{s}_0 = \langle 0, \quad 2, \quad 0 \rangle$$

y \ x	0	2	0
a	1	2	1
b	0	2	0
c	0	2	0

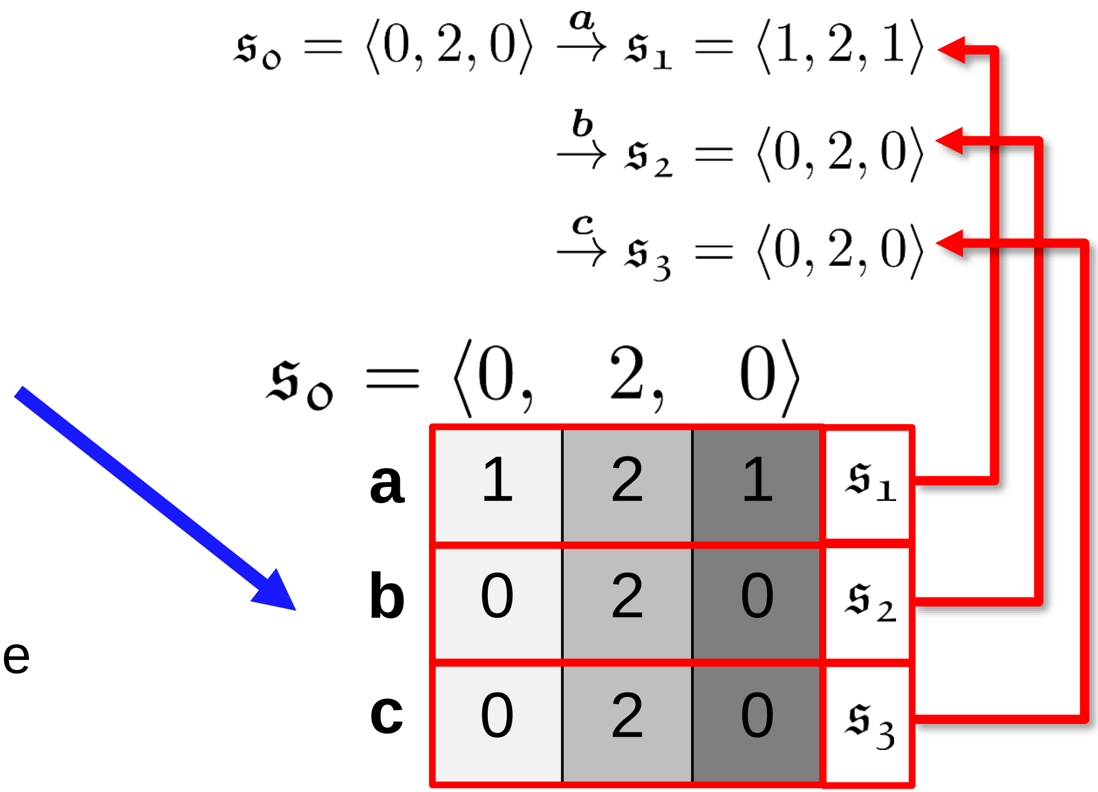
Optimized: transpose the $x \times y$ table to the $y \times x$ table according to the DFA-states of the source SFA-state

Parameterized transposition

- Speed up creating next SFA-states of each SFA-state

	a	b	c
0	1	0	0
1	1	0	2
2	2	2	2

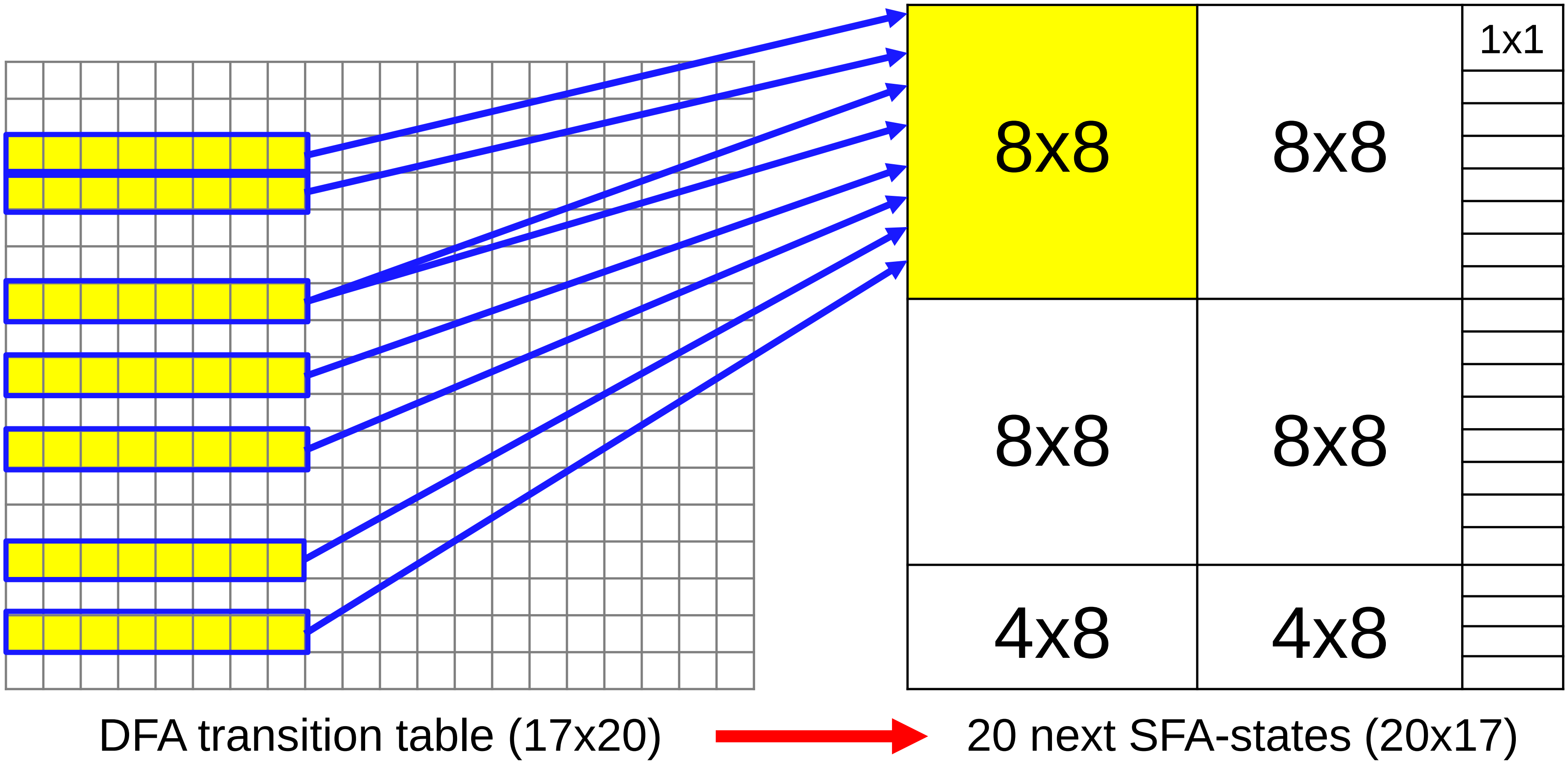
DFA transition table



Optimized: transpose the $x \times y$ table to the $y \times x$ table according to the DFA-states of the source SFA-state

Parameterized transposition (cont.)

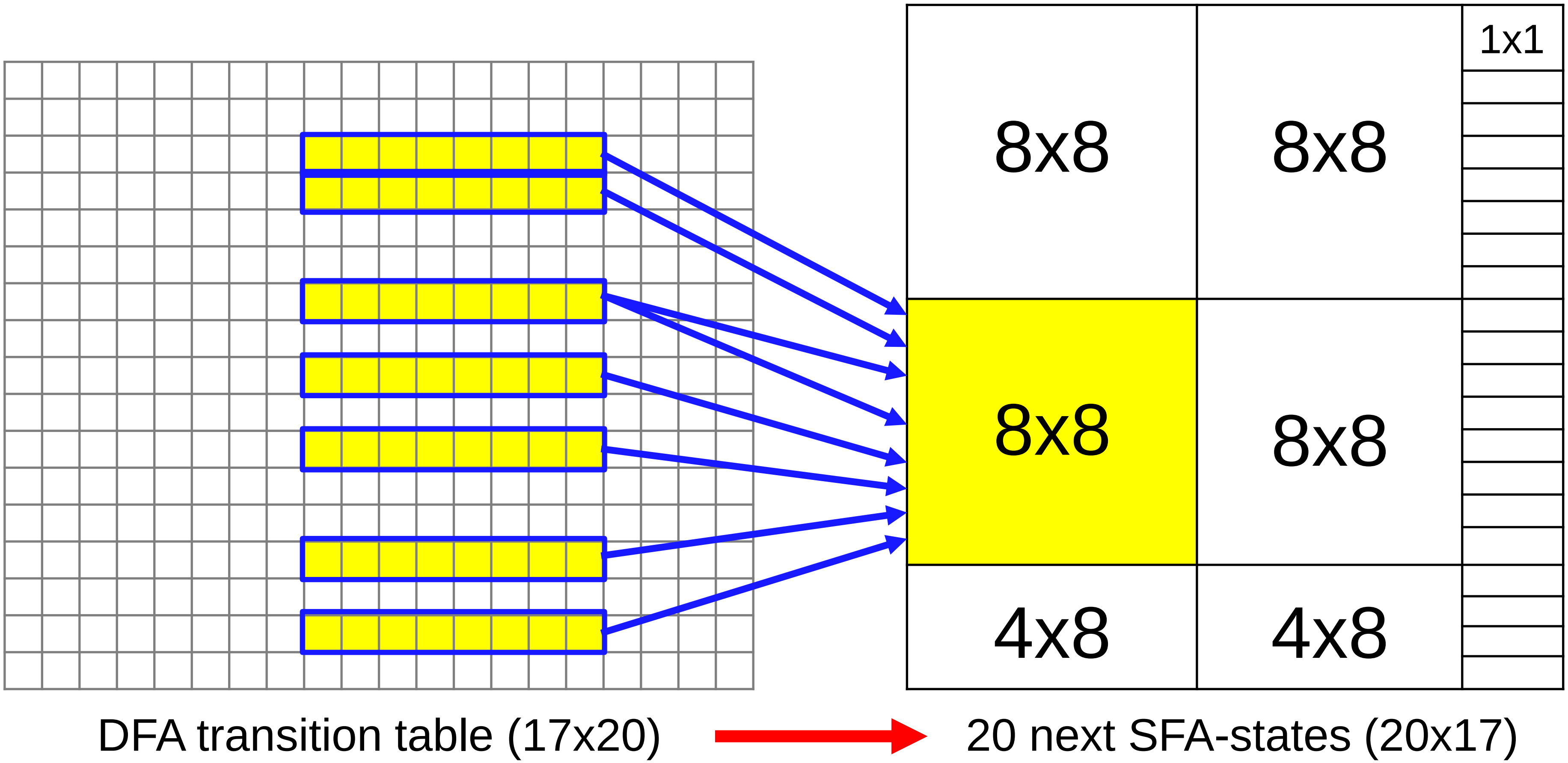
- Example transposed transition table
  - # DFA-states: 17, # symbols: 20



x86 SIMD-intrinsics-based transposition kernels

Parameterized transposition (cont.)

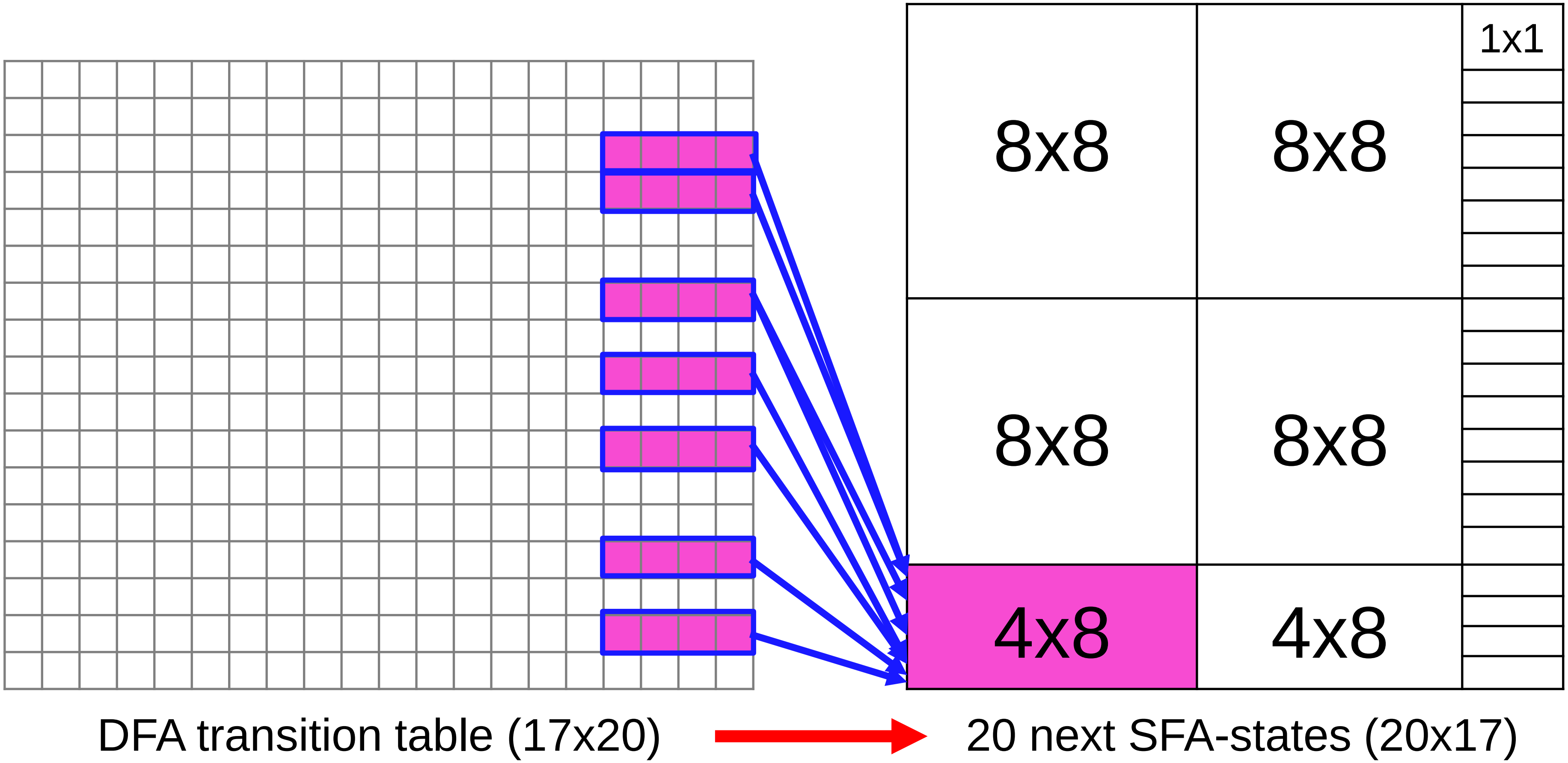
- Example transposed transition table
  - # DFA-states: 17, # symbols: 20



x86 SIMD-intrinsics-based transposition kernels

Parameterized transposition (cont.)

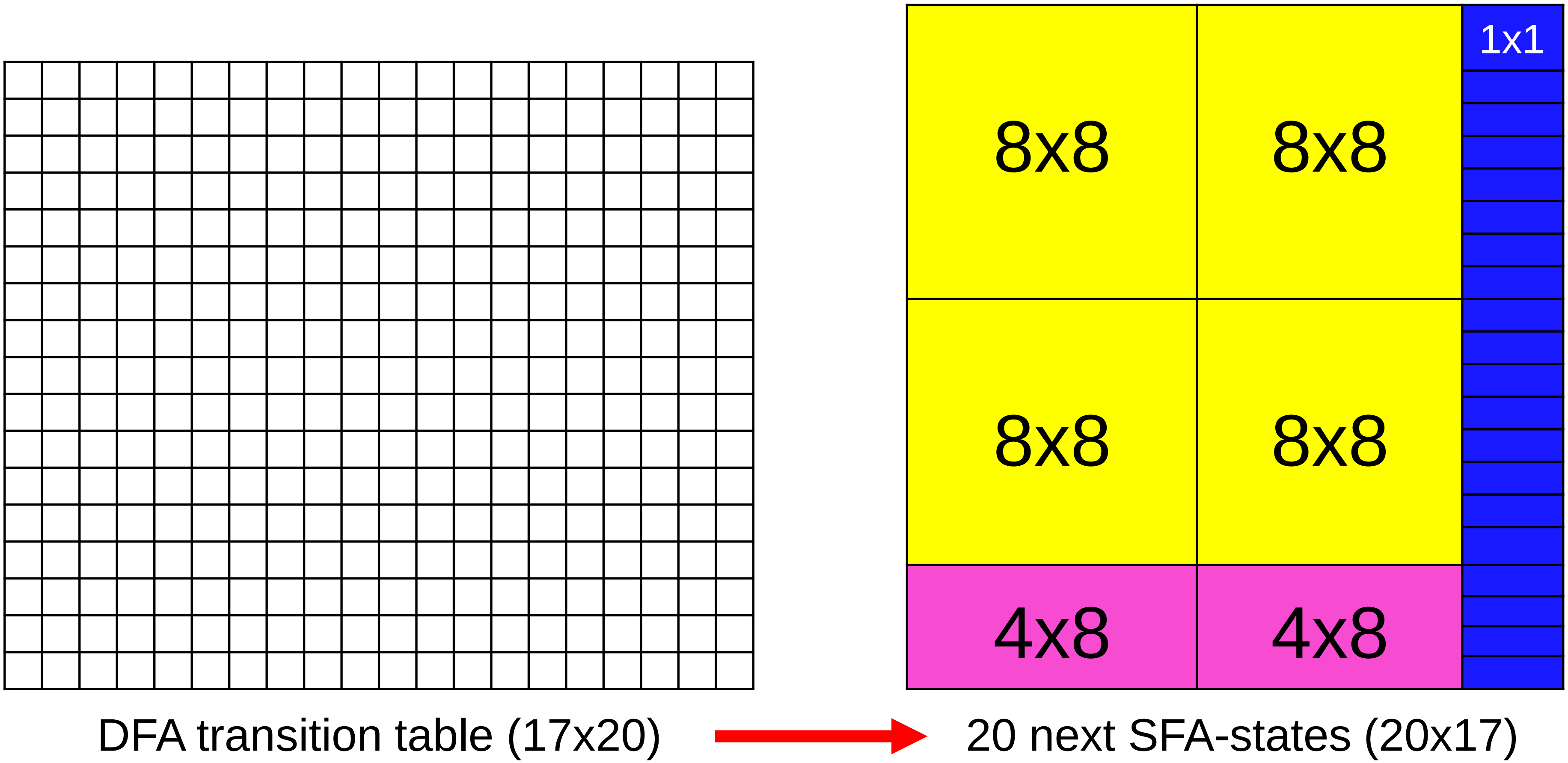
- Example transposed transition table
  - # DFA-states: 17, # symbols: 20



x86 SIMD-intrinsics-based transposition kernels

Parameterized transposition (cont.)

- Example transposed transition table
  - # DFA-states: 17, # symbols: 20

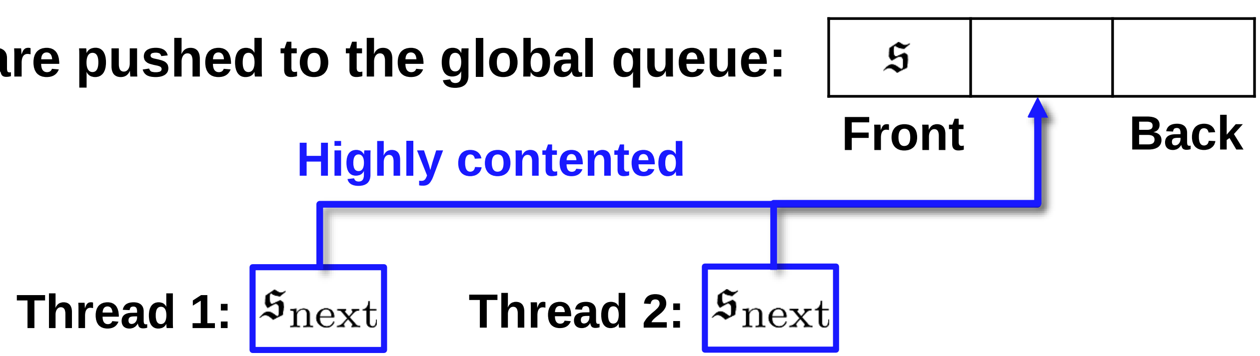


x86 SIMD-intrinsics-based transposition kernels

Work (SFA-state) distribution

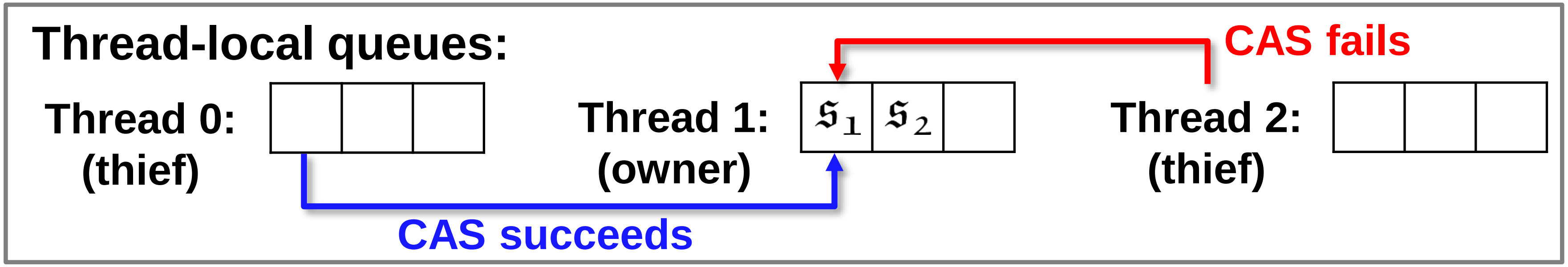
- **Observations:**
  1) The amount of work changes dynamically.
     - Few available states at the beginning, but soon all cores are saturated.
  2) Switching the work distribution scheme dynamically adapts to the changing load condition and reduces the cache-coherence overhead.
- **Scheme 1**: static distribution via a global queue:
  - **Advantage:** avoid coherence-overhead at front of the queue from work-stealing attempts of idle threads
  - Back of the queue is not contended because initially little work is available.

New SFA-states are pushed to the global queue:



Work (SFA-state) distribution (cont.)

- **Scheme 2**: dynamic distribution via thread-local queues
  - **Work-stealing**: steal work from the other's queue once the local queue is empty
    - Work will be popped exactly once by a thread because of lock-free synchronization using compare-and-swap (CAS) operation
  - **Advantage:** avoid coherence-overhead from the highly contended back of the global queue
  - Dequeuing SFA-states from other thread-local queues (work-stealing) makes front of the queue highly contended (cache coherence overhead) when little work is available



In-memory compression

- SFA-state compression mitigates **state explosion** problem

$$\mathfrak{s}_0 = \langle 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, \cdots, |Q| - 1 \rangle$$

27 KB per SFA-state

Compress

- Dictionary-based compression shows **high compression ratios** due to structural properties of FAs
  - FA-states tend to repeat in SFA-states

$$\mathfrak{s}_1 = \langle 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 0, 0, 0, 10, 0, 0, 0, 0, \cdots, |Q| - 1 \rangle$$

- Compression requires additional **costly** computation
- Initiate once a critical memory threshold is reached

In-memory compression (cont.)

- Mitigate **intractable** problem sizes
- Conduct SFA construction in three phases
  - **First phase**: construct an SFA with un-compressed SFA-states

$$\mathfrak{s}_0 = \langle 0,1,2\rangle \xrightarrow{b} \mathfrak{s}_1 = \langle 1,1,2\rangle$$
$$\xrightarrow{a} \mathfrak{s}_2 = \langle 0,0,2\rangle$$
$$\xrightarrow{c} \mathfrak{s}_3 = \langle 0,2,2\rangle$$

$\mathfrak{s}_0$	$\langle 0,1,2\rangle$
$\mathfrak{s}_1$	$\langle 1,1,2\rangle$
$\mathfrak{s}_2$	$\langle 0,0,2\rangle$
$\mathfrak{s}_3$	$\langle 0,2,2\rangle$

In-memory compression (cont.)

- Mitigate **intractable** problem sizes
- Conduct SFA construction in three phases
  - First phase: construct an SFA with un-compressed SFA-states
  - **Second phase**: compress all generated SFA-states once a critical memory threshold is reached

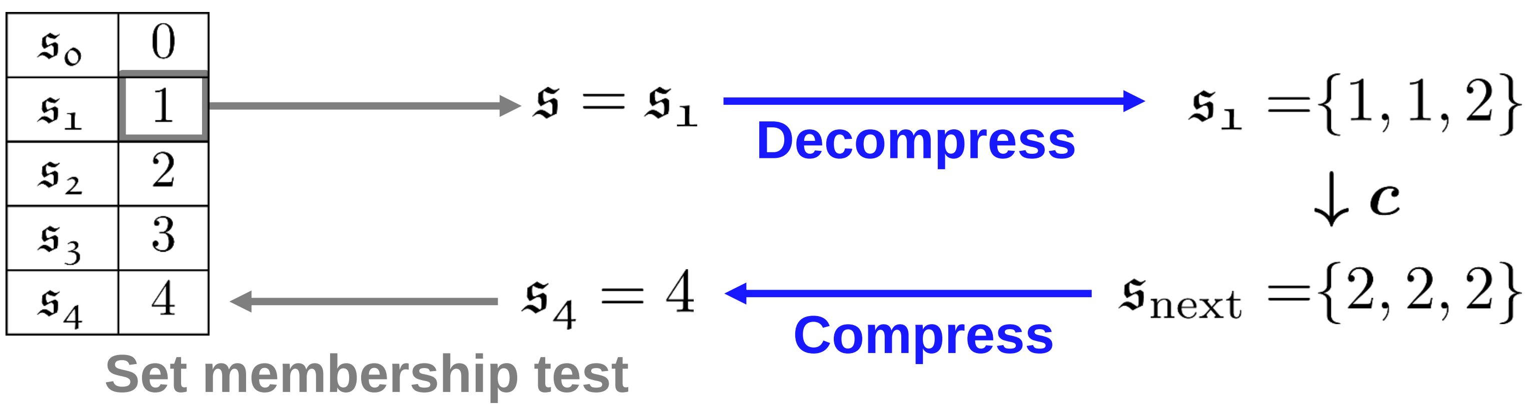
$\mathfrak{s}_0$	$\langle 0, 1, 2 \rangle$
$\mathfrak{s}_1$	$\langle 1, 1, 2 \rangle$
$\mathfrak{s}_2$	$\langle 0, 0, 2 \rangle$
$\mathfrak{s}_3$	$\langle 0, 2, 2 \rangle$

Dictionary-based lossless compression →

$\mathfrak{s}_0$	0
$\mathfrak{s}_1$	1
$\mathfrak{s}_2$	2
$\mathfrak{s}_3$	3

In-memory compression (cont.)

- Mitigate **intractable** problem sizes
- Conduct SFA construction in three phases
  - First phase: construct an SFA with un-compressed SFA-states
  - Second phase: compress all generated SFA-states once a critical memory threshold is reached
  - **Third phase**: resume SFA construction with compressed SFA-states



Experimental evaluation

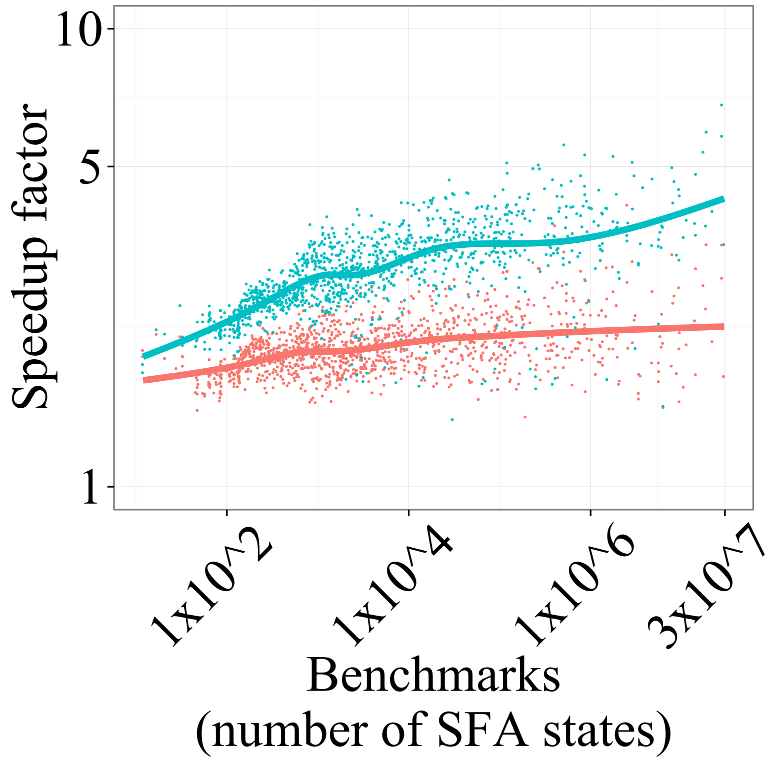
- Benchmarks: 1250 patterns from PROSITE protein database
  - Their minimal DFAs are generated by Grail+.
  - Exclude patterns take several days to convert to minimal DFAs.

- Proposed algorithm implemented in C11 using POSIX threads.
- Performance results are obtained by PAPI allows accesing hardware performance counters.

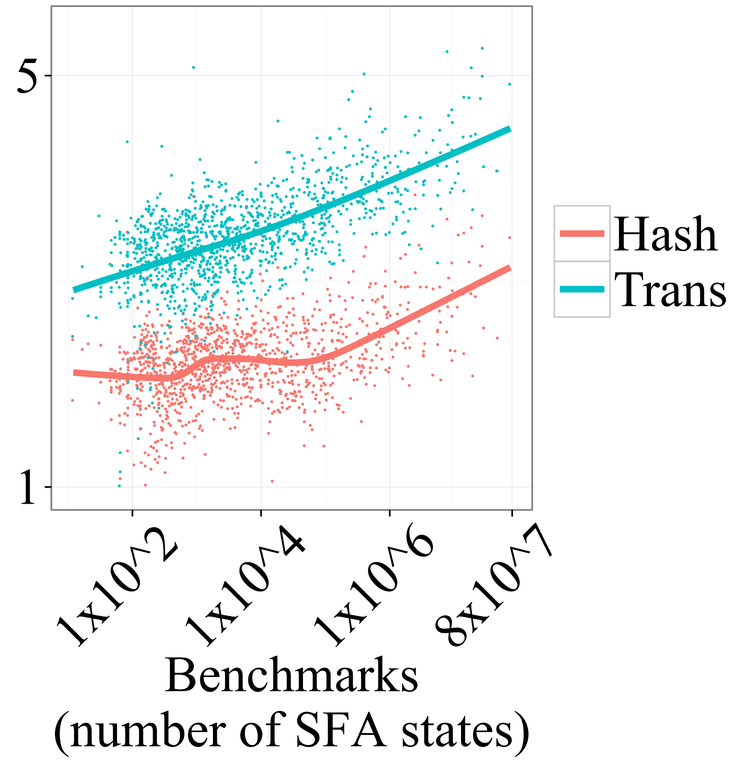
- Evaluation platforms:
  - 4-CPU (64 cores) AMD Opteron system
  - 2-CPU (44 cores, 2 hyperthreads per core) Intel Xeon Broadwell E5-2699 v4 system
  - Linux CentOS version 7

Experimental evaluation (cont.)

- Speedups of optimized sequential algorithm over the previous algorithm



On the AMD system



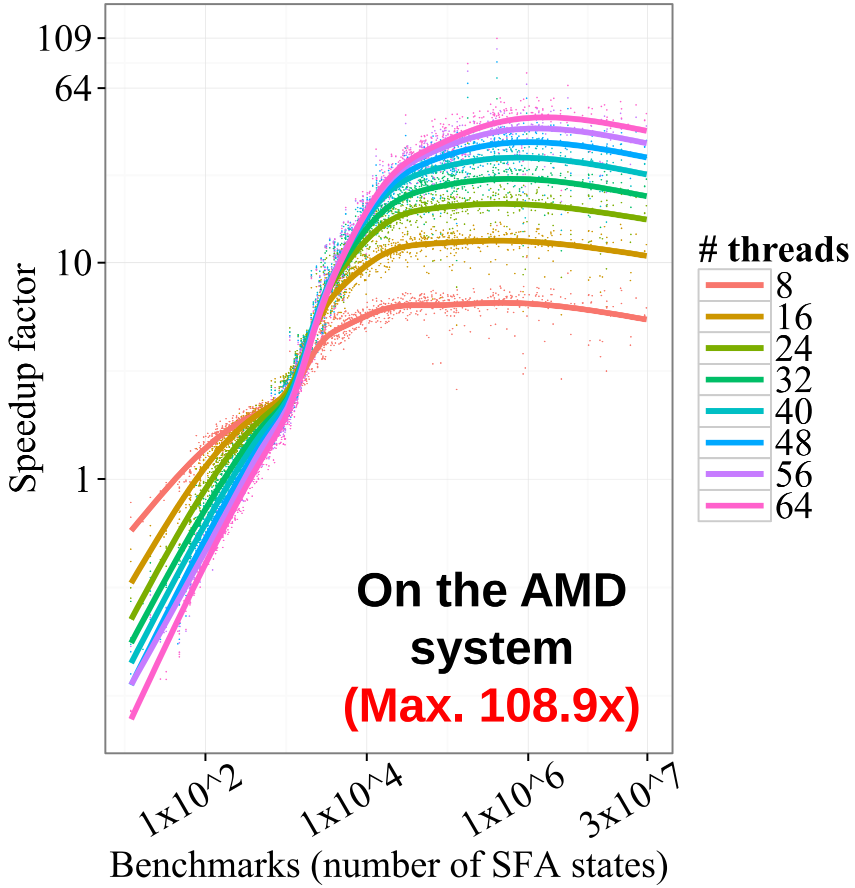
On the Intel system

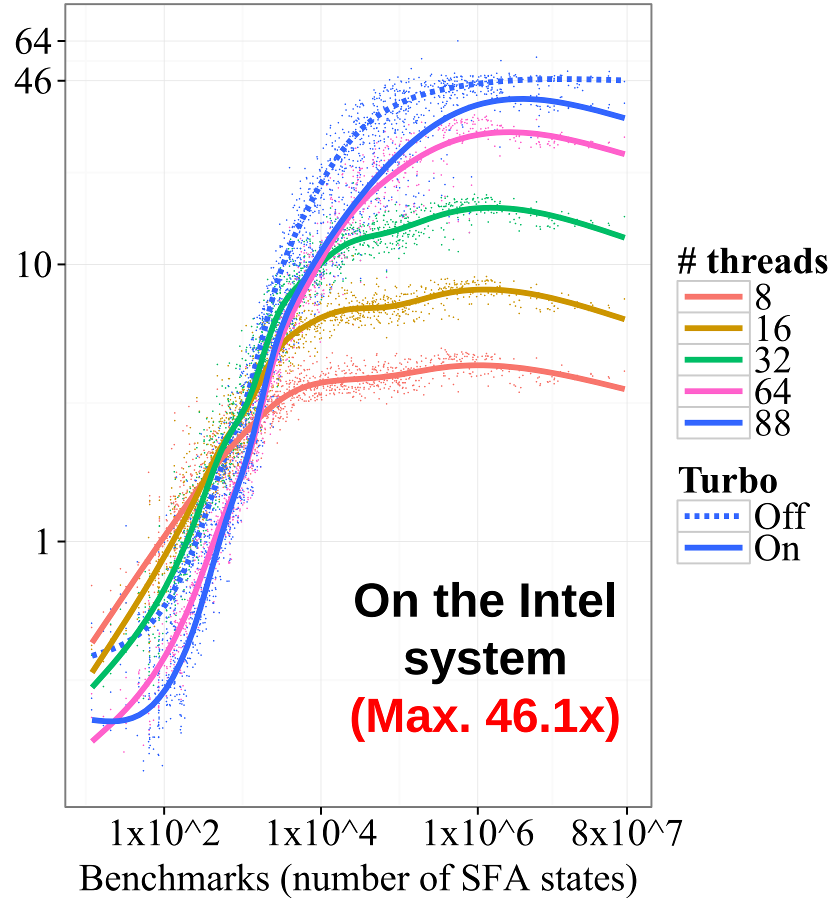
- **Hashing**: max 4.1x on AMD, 3.1x on Intel
- **Combination of hashing and transposition**:

  max 6.8x on AMD, 5.2x on Intel

Experimental evaluation (cont.)

- Speedups of parallelization
  - Based on our fastest sequential algorithm using hashing and parameterized transposition





Experimental evaluation (cont.)

- Performance and size comparison with and w/o compression
  - Six benchmarks on the Intel system (four benchmarks are intractable w/o compression and two benchmarks are added to compare them)
  - Set our memory manager's threshold to 200 GB to force compression of two tractable benchmarks

Number of States		w/o compr.		with compr.		Compr.
DFA	SFA	Size (MB)	Time (s)	Size (MB)	Time (s)	Ratio
2,557	74,624,878	381,632	42	12,622	1,015	30
2,980	40,956,096	244,098	28	13,172	436	19
6,132	17,795,082	436,478	n/a	12,153	824	18
6,419	20,559,280	527,880	n/a	15,495	1,212	17
6,549	47,076,417	1,233,214	n/a	29,610	2,700	21
7,025	23,975,400	673,709	n/a	20,106	1,641	17

Intractable w/o compression

Conclusion

- Introduced **fingerprints** and **hashing** to reduce state comparisons and set membership tests.
- **Parameterized transposition** of the transition table ensures cache locality of memory accesses.
- **Dynamic switch** from global work queue to **thread local queues** with work-stealing avoids contention of cache-lines at front and back of queue.
- Dynamically switch to **in-memory compression** of SFA-states once they cannot fit into the main memory.
- **Overall speedups** including fingerprint-based hashing, parameterized transposition and parallelization without compression are up to **312x** on AMD and **193x** on Intel.
- **Compression ratios** are up to **30** on the Intel system.

Acknowledgments

- This research was supported by:
  - the Austrian Science Fund (FWF) project I 1035N23
  - the Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF), funded by the Ministry of Science, ICT & Future Planning under grant NRF2015M3C4A7065522

Q&A

Thank you!