CONCUR 2021

Ismaël Jecker [1]

Nicolas Mazzocchi [2]

Petra Wolf [3]

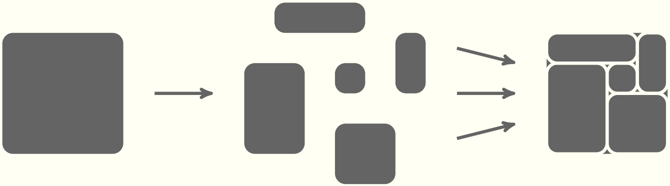
1 Institute of Science and Technology, Austria

2 IMDEA Software institute, Spain

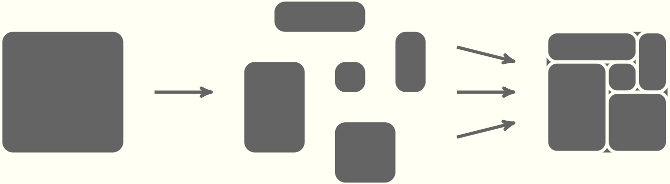
3 Universität Trier Informatikwissenschaften, Germany

Decomposing Permutation Automata

Divide and conquer



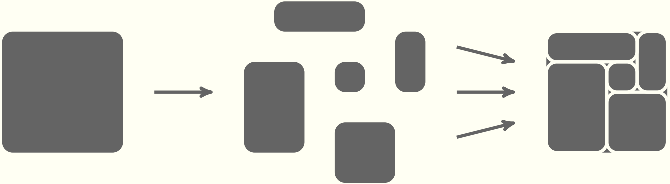
Divide and conquer



Compositionality

1. decomposition into smaller instances
2. .. that determine the original problem

Divide and conquer



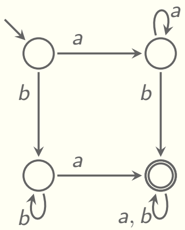
Compositionality

1. decomposition into smaller instances
2. .. that determine the original problem

Deterministic Finite state Automata

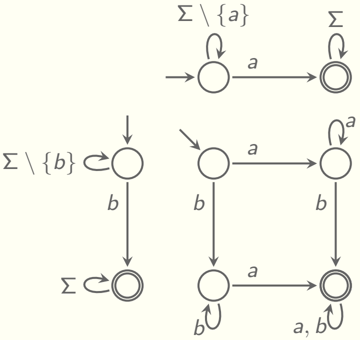
- construct automata with *less states*
- .. which preserve the original *language*

Example of decomposition



$\mathcal{A}$ is a DFA

Example of decomposition

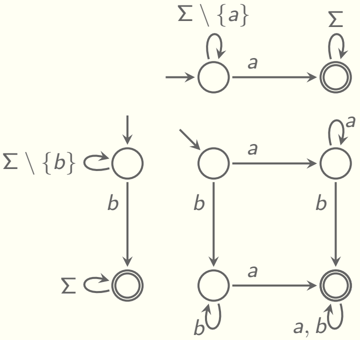


$\mathcal{A} = \mathcal{B}_1 \times \mathcal{B}_2$

Decomposition using 2 DFAs

- $|\mathcal{B}_1| < |\mathcal{A}|$ and $|\mathcal{B}_2| < |\mathcal{A}|$
- $L(\mathcal{A}) = L(\mathcal{B}_1) \cap L(\mathcal{B}_2)$

Example of decomposition



$\mathcal{A} = \mathcal{B}_1 \times \mathcal{B}_2$

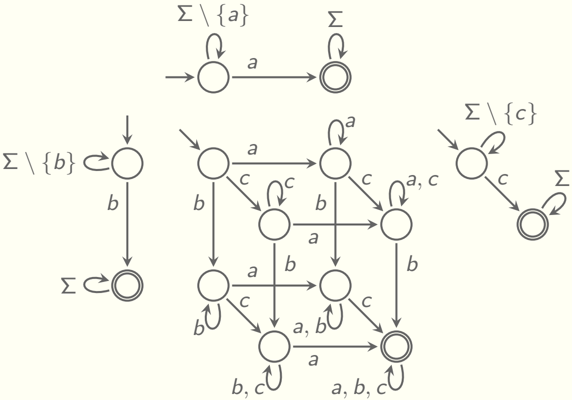
Decomposition using 2 DFAs

- $|\mathcal{B}_1| < |\mathcal{A}|$ and $|\mathcal{B}_2| < |\mathcal{A}|$
- $L(\mathcal{A}) = L(\mathcal{B}_1) \cap L(\mathcal{B}_2)$

Applications

- Hardware design of DFAs
- Tractability of Model-checking

Example of decomposition



$\mathcal{A} = \mathcal{B}_1 \times \mathcal{B}_2 \times \mathcal{B}_3$

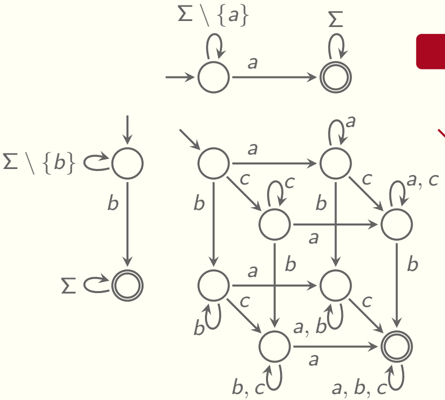
Decomposition using 3 DFAs

- $|\mathcal{B}_i| < |\mathcal{A}|$ for all i
- $L(\mathcal{A}) = \bigcap_i L(\mathcal{B}_i)$

Applications

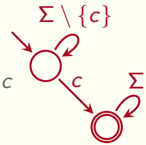
- Hardware design of DFAs
- Tractability of Model-checking

Example of decomposition



$\mathcal{A} = \mathcal{B}_1 \times \mathcal{B}_2 \times \mathcal{B}_3$

$\mathrm{LTL}_f : \Diamond c$



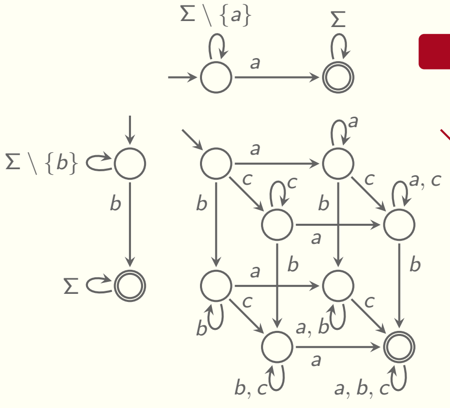
Decomposition using 3 DFAs

- $|\mathcal{B}_i| < |\mathcal{A}|$ for all i
- $L(\mathcal{A}) = \bigcap_i L(\mathcal{B}_i)$

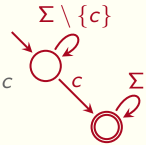
Applications

- Hardware design of DFAs
- Tractability of Model-checking

Example of decomposition



$\mathsf{LTL}_f : \Diamond c$



$\Psi = \bigwedge \Phi_i$

Decomposition using 3 DFAs

- $|\mathcal{B}_i| < |\mathcal{A}|$ for all i
- $L(\mathcal{A}) = \bigcap_i L(\mathcal{B}_i)$

Applications

- Hardware design of DFAs
- Tractability of Model-checking

General Setting

Definitions

Factors of DFA

The DFA $\mathcal{B}$ is a *factor* of the DFA $\mathcal{A}$ when:

- $|\mathcal{B}| < |\mathcal{A}|$ *smaller size* $<$
- $L(\mathcal{A}) \subseteq L(\mathcal{B})$ *coarser language* $\subset$

Definitions

Factors of DFA

The DFA $\mathcal{B}$ is a *factor* of the DFA $\mathcal{A}$ when:

- $|\mathcal{B}| < |\mathcal{A}|$ *smaller size* $<$
- $L(\mathcal{A}) \subseteq L(\mathcal{B})$ *coarser language* $\subseteq$

Compositionality of DFA

The DFA $\mathcal{A}$ is *k-factor composite* if there exists $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_k$ factors such that:

$$L(\mathcal{A}) = \bigcap_{i=1}^{k} L(\mathcal{B}_i)$$ rejectance preservation $=$

It is *composite* when it is k-factor composite for some k.

Decision algorithms

Decide whether $\mathcal{A}$ is composite (k is not given)

Decision algorithms

Decide whether $\mathcal{A}$ is composite (k is not given)

1. List all automata with less states than $\mathcal{A}$ ensures <

$\mathcal{B}_1$ $\mathcal{B}_2$ $\mathcal{B}_3$ $\mathcal{B}_4$ $\mathcal{B}_5$ $\mathcal{B}_6$ $\cdots$ $\mathcal{B}_n$

Decision algorithms

Decide whether $\mathcal{A}$ is composite (k is not given)

1. List all automata with less states than $\mathcal{A}$ — ensures $<$
2. Discard all $\mathcal{B}_i$ for which $L(\mathcal{A}) \not\subseteq L(\mathcal{B}_i)$ — ensures $\subseteq$

$\boxed{\mathcal{B}_1}$ $\boxed{\mathcal{B}_2}$ $\boxed{\mathcal{B}_3}$ $\boxed{\mathcal{B}_4}$ $\boxed{\mathcal{B}_5}$ $\boxed{\mathcal{B}_6}$ $\cdots$ $\boxed{\mathcal{B}_n}$

Decision algorithms

Decide whether $\mathcal{A}$ is composite (k is not given)

1. List all automata with less states than $\mathcal{A}$ ensures $<$
2. Discard all $\mathcal{B}_i$ for which $L(\mathcal{A}) \not\subseteq L(\mathcal{B}_i)$ ensures $\subset$

$$L\left(\boxed{\mathcal{B}_1} \times \boxed{\mathcal{B}_4} \times \boxed{\mathcal{B}_6} \times \cdots \times \boxed{\mathcal{B}_n} \right) \supseteq L(\mathcal{A})$$

Decision algorithms

Decide whether $\mathcal{A}$ is composite (k is not given)

1. List all automata with less states than $\mathcal{A}$ — ensures $<$
2. Discard all $\mathcal{B}_i$ for which $L(\mathcal{A}) \not\subseteq L(\mathcal{B}_i)$ — ensures $\subseteq$
3. Check that $\bigcap_i L(\mathcal{B}_i) = L(\mathcal{A})$ — ensures $=$

$$L\left(\boxed{\mathcal{B}_1} \times \boxed{\mathcal{B}_4} \times \boxed{\mathcal{B}_6} \times \cdots \times \boxed{\mathcal{B}_n} \right) = L(\mathcal{A})$$

Decision algorithms

Decide whether $\mathcal{A}$ is composite (k is not given)

1. List all automata with less states than $\mathcal{A}$ — ensures $<$
2. Discard all $\mathcal{B}_i$ for which $L(\mathcal{A}) \not\subseteq L(\mathcal{B}_i)$ — ensures $\subseteq$
3. Check that $\bigcap_i L(\mathcal{B}_i) = L(\mathcal{A})$ — ensures $=$

$$L\left(\boxed{\mathcal{B}_1} \times \boxed{\mathcal{B}_4} \times \boxed{\mathcal{B}_6} \times \cdots \times \boxed{\mathcal{B}_n} \right) = L(\mathcal{A})$$

Theorem of Kupferman and Mosheiff in MFCS'13 and J. Inf. Comput. 2015 [1]

- *Compositionality is in* ExpSpace *and* NLogSpace*-hard for DFAs*

Decision algorithms

Decide whether $\mathcal{A}$ is composite (k is not given)

1. List all automata with less states than $\mathcal{A}$ — ensures $<$
2. Discard all $\mathcal{B}_i$ for which $L(\mathcal{A}) \nsubseteq L(\mathcal{B}_i)$ — ensures $\subset$
3. Check that $\bigcap_i L(\mathcal{B}_i) = L(\mathcal{A})$ — ensures $=$

$$L\left(\boxed{\mathcal{B}_1} \times \boxed{\mathcal{B}_4} \times \boxed{\mathcal{B}_6} \times \cdots \times \boxed{\mathcal{B}_n} \right) = L(\mathcal{A})$$

Theorem of Kupferman and Mosheiff in MFCS'13 and J. Inf. Comput. 2015 [1]

- *Compositionality is in* EXPSPACE *and* NLOGSPACE*-hard for DFAs* — *best known bounds*

Decision algorithms

Decide whether $\mathcal{A}$ is k-factor composite

1. ~~List all~~ **Guess k** automata smaller than $\mathcal{A}$ — ensures $<$
2. Discard all $\mathcal{B}_i$ for which $L(\mathcal{A}) \not\subseteq L(\mathcal{B}_i)$ — ensures $\subset$
3. Check that $\bigcap_i L(\mathcal{B}_i) = L(\mathcal{A})$ — ensures $=$

$$L\left(\boxed{\mathcal{B}_1} \times \boxed{\mathcal{B}_4} \times \boxed{\mathcal{B}_6} \times \cdots \times \boxed{\mathcal{B}_k} \right) = L(\mathcal{A})$$

Theorem of Kupferman and Mosheiff in MFCS'13 and J. Inf. Comput. 2015 [1]

- *Compositionality is in* EXPSPACE *and* NLOGSPACE*-hard for DFAs* — *best known bounds*

Decision algorithms

Decide whether $\mathcal{A}$ is k-factor composite

1. ~~List all~~ **Guess k** automata smaller than $\mathcal{A}$ — ensures $<$
2. Discard all $\mathcal{B}_i$ for which $L(\mathcal{A}) \not\subseteq L(\mathcal{B}_i)$ — ensures $\subset$
3. Check that $\bigcap_i L(\mathcal{B}_i) = L(\mathcal{A})$ — ensures $=$

$$L\left(\boxed{\mathcal{B}_1} \times \boxed{\mathcal{B}_4} \times \boxed{\mathcal{B}_6} \times \cdots \times \boxed{\mathcal{B}_k} \right) = L(\mathcal{A})$$

Theorems

- *Compositionality is in* $\textsc{ExpSpace}$ *and* $\textsc{NLogSpace}$*-hard for DFAs* — *best known bounds*
- *It is in* $\textsc{PSpace}$ *once parameterized to have at most k-factors*

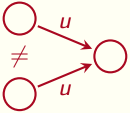
Contributions

Automata subclasses

Permutation automata

The transition function denotes a permutation on states for every input

forbidden pattern

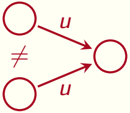


Automata subclasses

Permutation automata

The transition function denotes a permutation on states for every input

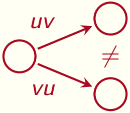
forbidden pattern



Commutative automata

The transition function does not depends on the input order

forbidden pattern

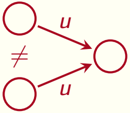


Automata subclasses

Permutation automata

The transition function denotes a permutation on states for every input

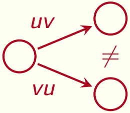
forbidden pattern



Commutative automata

The transition function does not depends on the input order

forbidden pattern



Kupferman and Mosheiff in [1]

(commutative) permutation DFA is composite

$\Updownarrow$

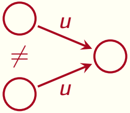
a decomposition keeps structural properties

Automata subclasses

Permutation automata

The transition function denotes a permutation on states for every input

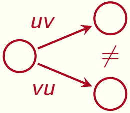
forbidden pattern



Commutative automata

The transition function does not depends on the input order

forbidden pattern



Kupferman and Mosheiff in [1]

(commutative) permutation DFA is composite

$\Updownarrow$

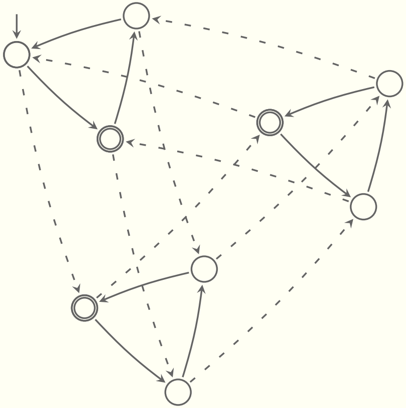
a decomposition keeps structural properties

Jecker, Mazzocchi and Wolf

1. Simpler structural characterizations
   - .. implying better complexity results
2. Lower bound on minimal factor numbers

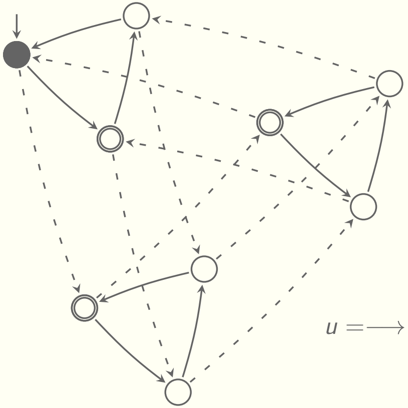
Commutative permutation DFAs

Visiting states using word iteration



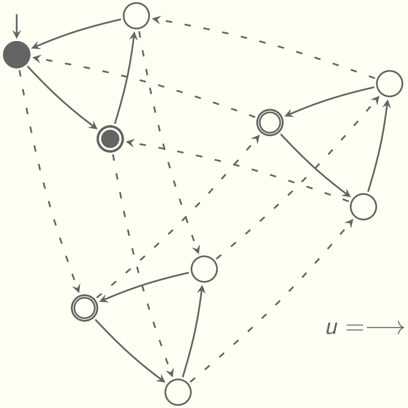
- permutation automaton
- commutative automaton

Visiting states using word iteration



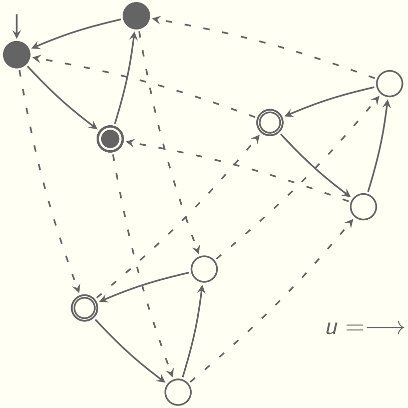
- permutation automaton
- commutative automaton

Visiting states using word iteration



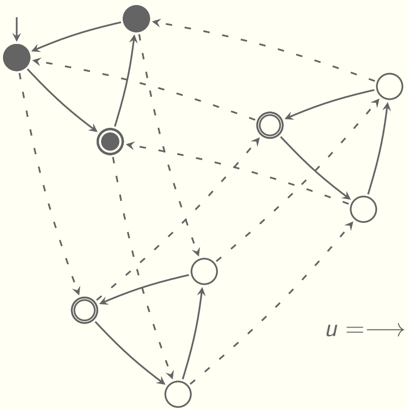
- permutation automaton
- commutative automaton

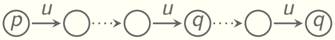
Visiting states using word iteration



- permutation automaton
- commutative automaton

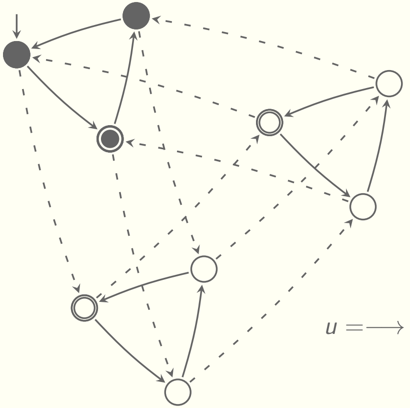
Visiting states using word iteration

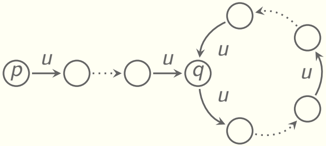




- permutation automaton
- commutative automaton

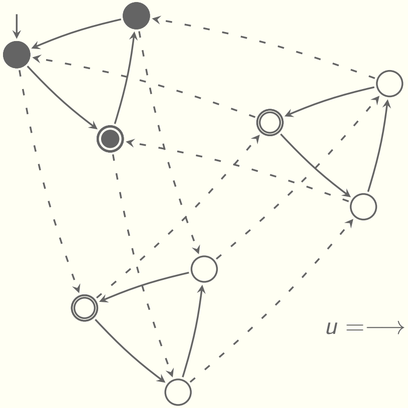
Visiting states using word iteration

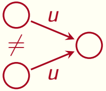


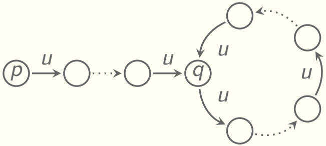


- permutation automaton
- commutative automaton

Visiting states using word iteration

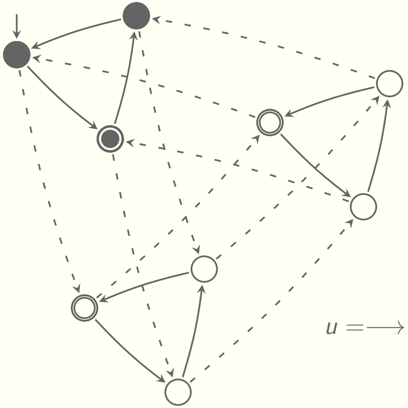


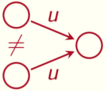


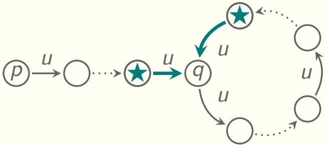


- permutation automaton
- commutative automaton

Visiting states using word iteration

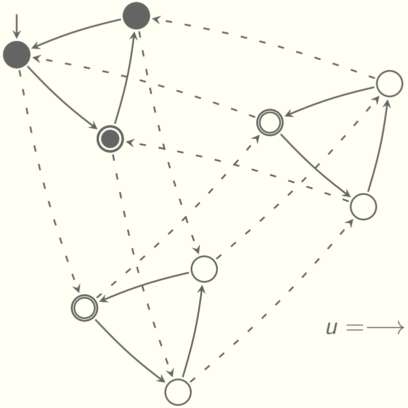




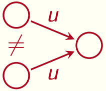


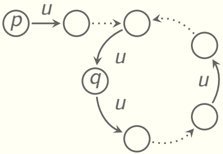
- permutation automaton
- commutative automaton

Visiting states using word iteration

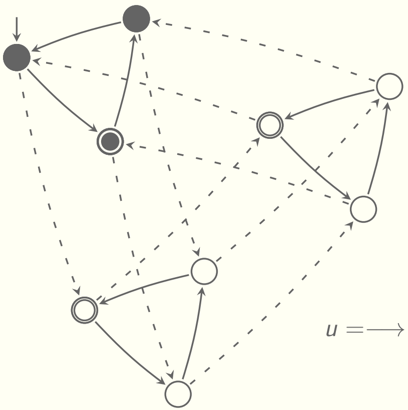


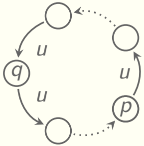
- permutation automaton
- commutative automaton





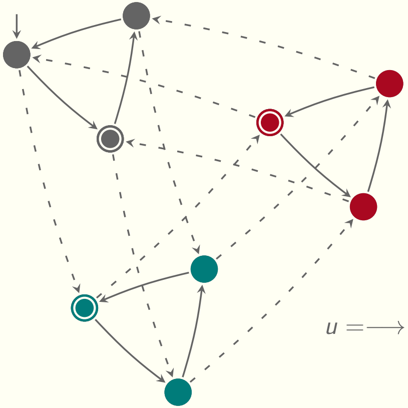
Visiting states using word iteration





- permutation automaton
- commutative automaton

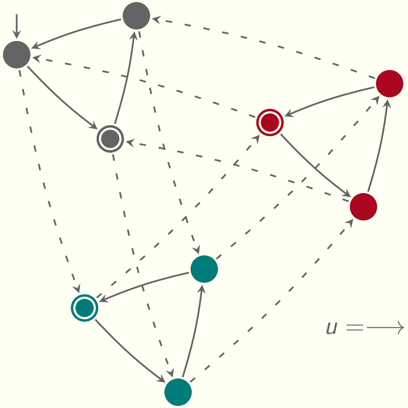
Visiting states using word iteration

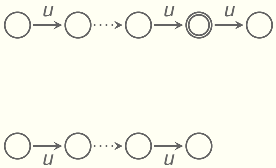


- permutation automaton
- commutative automaton

- $[p]_u = \{\delta(p, u^n) : n \in \mathbb{N}\}$

Visiting states using word iteration

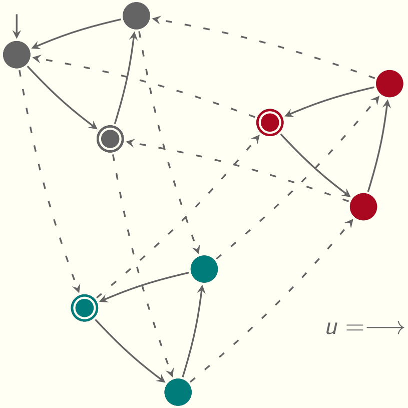


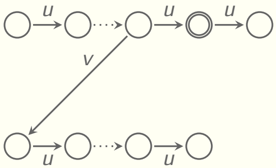


- permutation automaton
- commutative automaton

- $[p]_u = \{\delta(p, u^n) : n \in \mathbb{N}\}$

Visiting states using word iteration

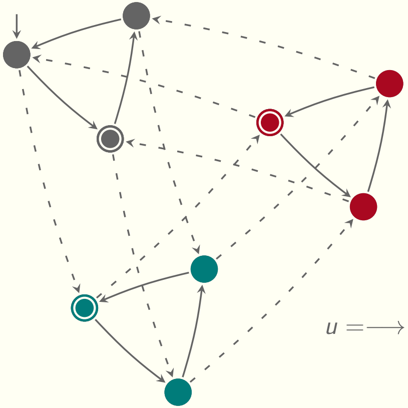


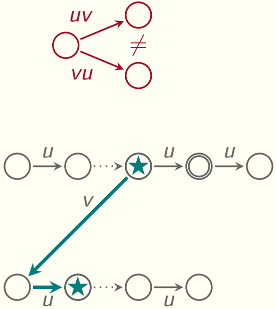


- permutation automaton
- commutative automaton

- $[p]_u = \{\delta(p, u^n) : n \in \mathbb{N}\}$

Visiting states using word iteration

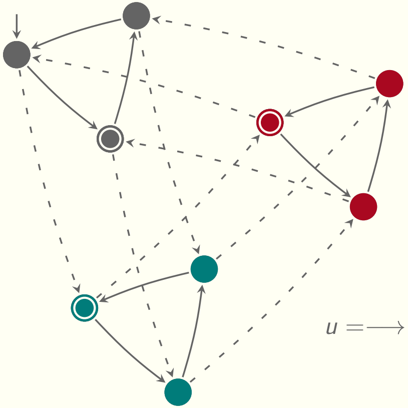


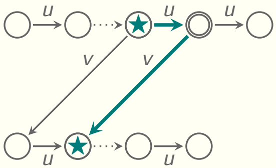


- permutation automaton
- commutative automaton

- $[p]_u = \{\delta(p, u^n) : n \in \mathbb{N}\}$

Visiting states using word iteration

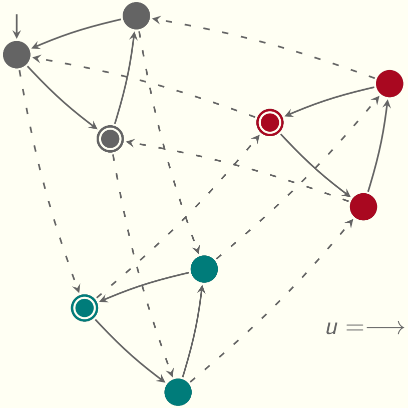


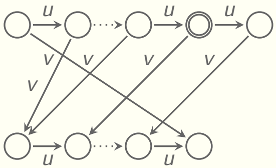


- permutation automaton
- commutative automaton

- $[p]_u = \{\delta(p, u^n) : n \in \mathbb{N}\}$

Visiting states using word iteration

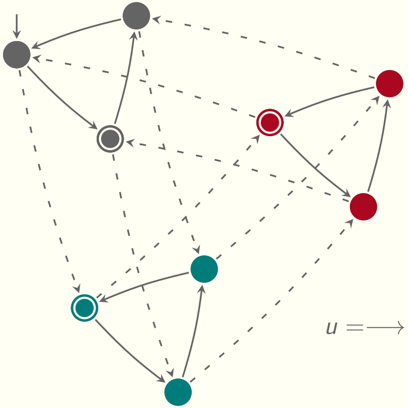


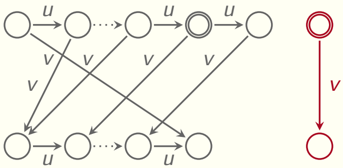


- permutation automaton
- commutative automaton

- $[p]_u = \{\delta(p, u^n) : n \in \mathbb{N}\}$
- $p \xrightarrow{a} q \iff [p]_u \xrightarrow{a} [q]_u$

Visiting states using word iteration

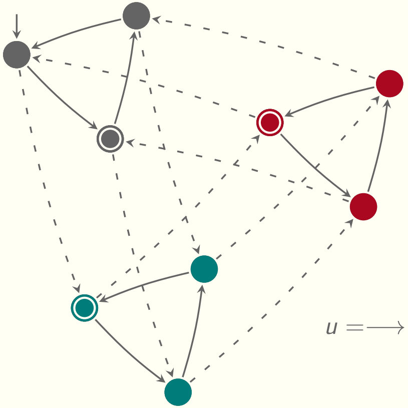




- permutation automaton
- commutative automaton

- $[p]_u = \{\delta(p, u^n) : n \in \mathbb{N}\}$
- $p \xrightarrow{a} q \iff [p]_u \xrightarrow{a} [q]_u$

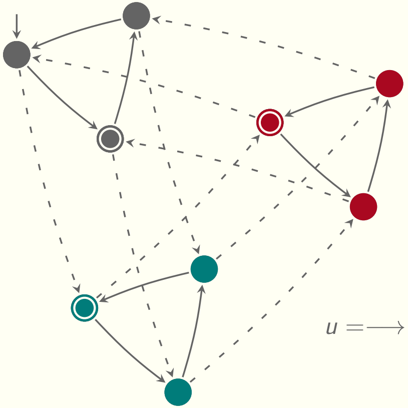
Visiting states using word iteration

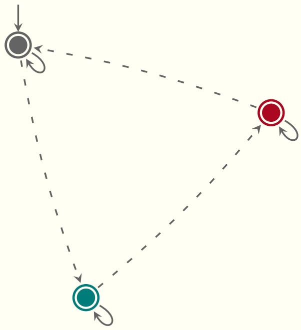


- permutation automaton
- commutative automaton

- $[p]_u = \{\delta(p, u^n) : n \in \mathbb{N}\}$
- $p \xrightarrow{a} q \iff [p]_u \xrightarrow{a} [q]_u$

Visiting states using word iteration





- permutation automaton
- commutative automaton

- **effective** merge of states $\Rightarrow$ ensures $<$
- relaxed acceptance $\Rightarrow$ ensures $\subset$

Characterization by word coverage

Word u that covers $p \in \neg F$

- $\left|[p]_u\right| > 1$
- $[p]_u \cap F = \varnothing$

Characterization by word coverage

Word u that covers $p \in \neg F$

- $\left|[p]_u\right| > 1$
- $[p]_u \cap F = \varnothing$

Key equivalence ⏩ sketch

a commutative permutation DFA is k-factor composite

$\Updownarrow$

k words suffice to cover all non-finals states

Characterization by word coverage

Word u that covers $p \in \neg F$

- $\left|[p]_u\right| > 1$
- $[p]_u \cap F = \varnothing$

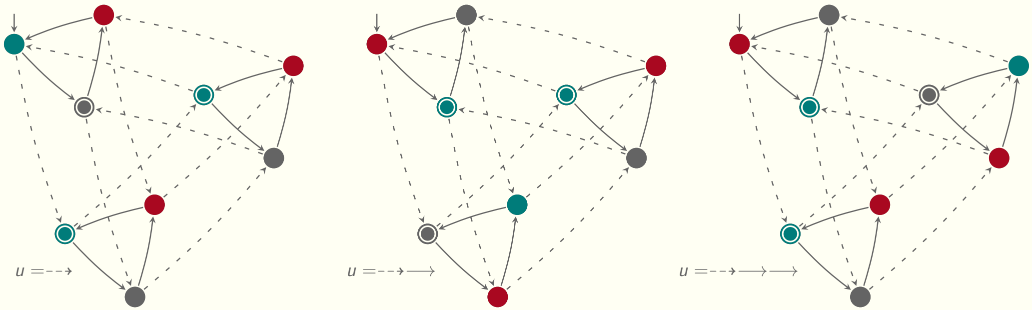
Key equivalence

⏩ sketch

a commutative permutation DFA is k-factor composite

$\Updownarrow$

k words suffice to cover all non-finals states



Characterization by word coverage

Word u that covers $p \in \neg F$

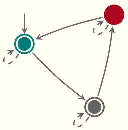
- $\left|[p]_u\right| > 1$
- $[p]_u \cap F = \varnothing$

Key equivalence ⏩ sketch

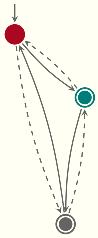
a commutative permutation DFA is k-factor composite

$\Updownarrow$

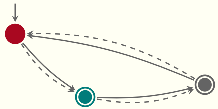
k words suffice to cover all non-finals states



$u = \dashrightarrow$



$u = \dashrightarrow\longrightarrow$



$u = \dashrightarrow\longrightarrow\longrightarrow$

Characterization by word coverage

Word u that covers $p \in \neg F$

- $\big|[p]_u\big| > 1$
- $[p]_u \cap F = \varnothing$

Key equivalence ⏩ sketch

a commutative permutation DFA is k-factor composite
$\Updownarrow$
k words suffice to cover all non-finals states

Deciding compositionality of commutative permutation DFAs (k is not given)

1. For each rejecting state $p \in \neg F$
   1.1 Guess $p' \neq p$ defining $\mathcal{A}$: $p \xrightarrow{u} p'$
   1.2 Check *on-the-fly* $\delta(p, u^i) \in \neg F$ for all $i \in \{1, \ldots, |\mathcal{A}|\}$

Characterization by word coverage

Word u that covers $p \in \neg F$

- $\left|[p]_u\right| > 1$
- $[p]_u \cap F = \varnothing$

Key equivalence ⏩ sketch

a commutative permutation DFA is k-factor composite

$\Updownarrow$

k words suffice to cover all non-finals states

Deciding compositionality of commutative permutation DFAs (k is not given)

1. For each rejecting state $p \in \neg F$
   1.1 Guess $p' \neq p$ defining $\mathcal{A}$: $p \xrightarrow{u} p'$
   1.2 Check *on-the-fly* $\delta(p, u^i) \in \neg F$ for all $i \in \{1, \dots, |\mathcal{A}|\}$

Theorem

- *Compositionality is in* NLOGSPACE *for commutative permutation DFAs*

Characterization by word coverage

Word u that covers $p \in \neg F$

- $\left|[p]_u\right| > 1$
- $[p]_u \cap F = \varnothing$

Key equivalence

⏩ sketch

a commutative permutation DFA is k-factor composite
$\Updownarrow$
k words suffice to cover all non-finals states

Deciding k-factor compositionality of commutative permutation DFAs

1. **Guess** $W = \{(p_1, p_1'), (p_2, p_2'), \ldots, (p_k, p_k') : p_j \neq p_j'\}$
2. For each rejecting state $p \in \neg F$

   2.1 Guess $(p_j, p_j') \in W$ defining $\mathcal{A}$: $p_j \xrightarrow{u_j} p_j'$

   2.2 Check *on-the-fly* $\delta(p, u_j^i) \in \neg F$ for all $i \in \{1, \ldots, |\mathcal{A}|\}$

Theorem

- *Compositionality is in* $\mathrm{NL{\small OG}S{\small PACE}}$ *for commutative permutation DFAs*

Characterization by word coverage

Word u that covers $p \in \neg F$

- $\big|[p]_u\big| > 1$
- $[p]_u \cap F = \varnothing$

Key equivalence ⏩ sketch

a commutative permutation DFA is k-factor composite
$\Updownarrow$
k words suffice to cover all non-finals states

Deciding k-factor compositionality of commutative permutation DFAs

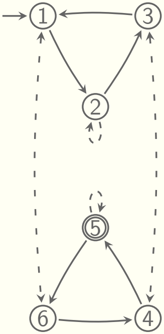
1. **Guess** $W = \{(p_1, p'_1), (p_2, p'_2), \ldots, (p_k, p'_k) : p_j \neq p'_j\}$
2. For each rejecting state $p \in \neg F$
   2.1 Guess $(p_j, p'_j) \in W$ defining $\mathcal{A}$: $p_j \xrightarrow{u_j} p'_j$
   2.2 Check *on-the-fly* $\delta(p, u^i_j) \in \neg F$ for all $i \in \{1, \ldots, |\mathcal{A}|\}$

Theorems

- *Compositionality is in* NLOGSPACE *for commutative permutation DFAs*
- *It is* NP-COMPLETE *once parameterized to have at most k-factors*

Permutation DFAs

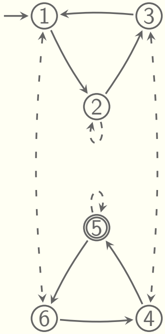
Generalizing word coverage



- permutation automaton
- **NOT** commutative

- define (or induce) group states
- .. while being consistent with transitions

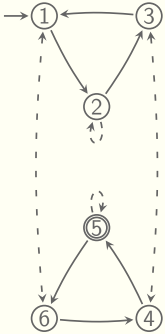
Generalizing word coverage

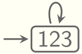


- permutation automaton
- **NOT** commutative

- define (or induce) groups of states
- .. while being consistent with transitions

Generalizing word coverage

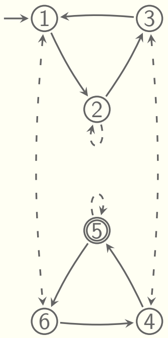




- permutation automaton
- **NOT** commutative

- define (or induce) groups of states
- .. while being consistent with transitions

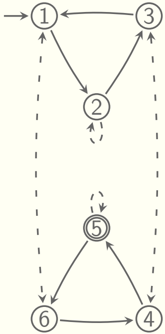
Generalizing word coverage

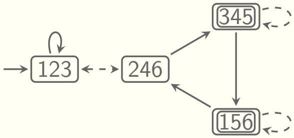


- permutation automaton
- **NOT** commutative

- define (or induce) groups of states
- .. while being consistent with transitions

Generalizing word coverage

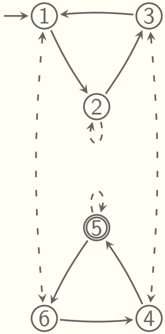


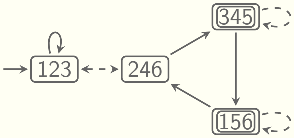


- permutation automaton
- **NOT** commutative

- define (or induce) groups of states
- .. while being consistent with transitions

Generalizing word coverage

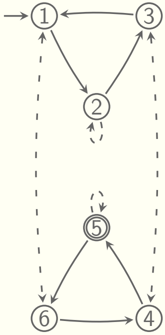


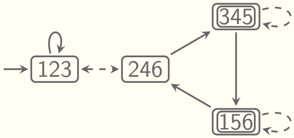


- permutation automaton
- **NOT** commutative

- $P, P' \in \{\delta(S, u) : u \in \Sigma^*\}$
- $p \xrightarrow{a} p' \iff P \xrightarrow{a} P'$ where $p \in P, p' \in P'$

Generalizing word coverage





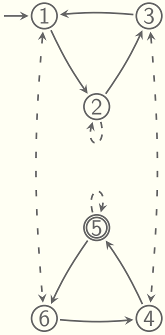
Factor? Usefull?

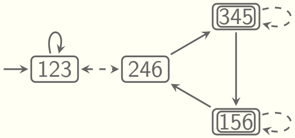
- $<$
- $\subset$
- $=$

- permutation automaton
- **NOT** commutative

- $P, P' \in \{\delta(S, u) : u \in \Sigma^*\}$
- $p \xrightarrow{a} p' \iff P \xrightarrow{a} P'$ where $p \in P, p' \in P'$

Generalizing word coverage





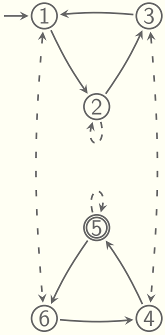
Factor? Usefull?

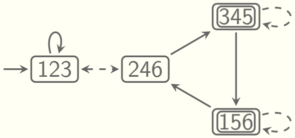
- $<$ needs to be checked
- $\subset$
- $=$

- permutation automaton
- **NOT** commutative

- $P, P' \in \{\delta(S, u) : u \in \Sigma^*\}$
- $p \xrightarrow{a} p' \iff P \xrightarrow{a} P'$ where $p \in P, p' \in P'$

Generalizing word coverage





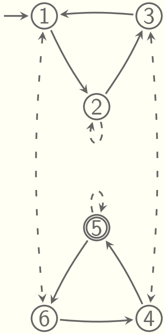
Factor? Usefull?

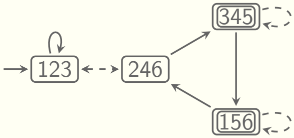
- needs to be checked
- holds by construction
-

- permutation automaton
- **NOT** commutative

- $P, P' \in \{\delta(S, u) : u \in \Sigma^*\}$
- $p \xrightarrow{a} p' \iff P \xrightarrow{a} P'$ where $p \in P, p' \in P'$

Generalizing word coverage





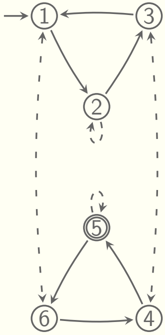
Factor? Usefull?

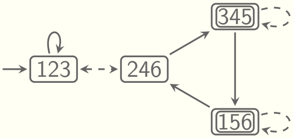
- needs to be checked
- holds by construction
- unfulfilled but..

- permutation automaton
- **NOT** commutative

- $P, P' \in \{\delta(S, u) : u \in \Sigma^*\}$
- $p \xrightarrow{a} p' \iff P \xrightarrow{a} P'$ where $p \in P, p' \in P'$

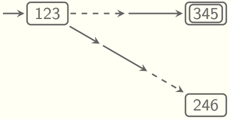
Generalizing word coverage





Factor? Usefull?

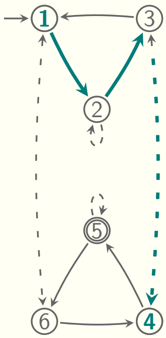
- $<$ needs to be checked
- $\subset$ holds by construction
- $=$ unfulfilled but..

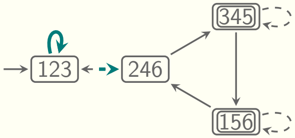


- permutation automaton
- **NOT** commutative

- $P, P' \in \{\delta(S, u) : u \in \Sigma^*\}$
- $p \xrightarrow{a} p' \iff P \xrightarrow{a} P'$ where $p \in P, p' \in P'$

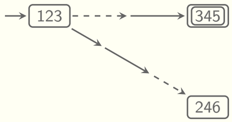
Generalizing word coverage





Factor? Usefull?

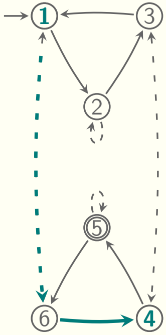
- $<$ needs to be checked
- $\subset$ holds by construction
- $=$ unfulfilled but..

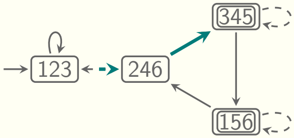


- permutation automaton
- **NOT** commutative

- $P, P' \in \{\delta(S, u) : u \in \Sigma^*\}$
- $p \xrightarrow{a} p' \iff P \xrightarrow{a} P'$ where $p \in P, p' \in P'$

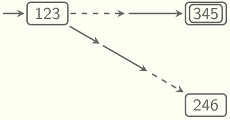
Generalizing word coverage





Factor? Usefull?

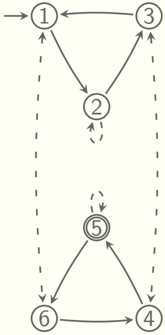
- $<$ needs to be checked
- $\subset$ holds by construction
- $=$ unfulfilled but..

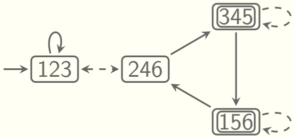


- permutation automaton
- **NOT** commutative

- $P, P' \in \{\delta(S,u) : u \in \Sigma^*\}$
- $p \xrightarrow{a} p' \iff P \xrightarrow{a} P'$ where $p \in P, p' \in P'$

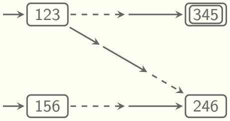
Generalizing word coverage





Factor? Usefull?

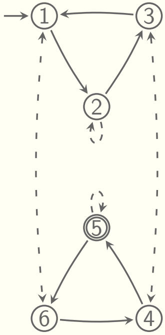
- $<$ needs to be checked
- $\subset$ holds by construction
- $=$ unfulfilled but.. it's fine

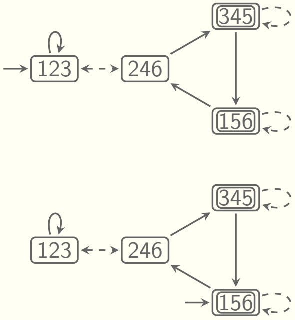


- permutation automaton
- **NOT** commutative

- $P, P' \in \{\delta(S,u) : u \in \Sigma^*\}$
- $p \xrightarrow{a} p' \iff P \xrightarrow{a} P'$ where $p \in P, p' \in P'$

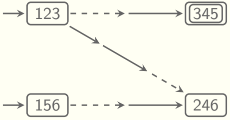
Generalizing word coverage





Factor? Usefull?

- needs to be checked
- holds by construction
- unfulfilled but.. it's fine



- permutation automaton
- **NOT** commutative

- $P, P' \in \{\delta(S,u) : u \in \Sigma^*\}$
- $p \xrightarrow{a} p' \iff P \xrightarrow{a} P'$ where $p \in P, p' \in P'$

Characterization by set coverage

Set S that covers $p \in \neg F$

- $\left|\{\delta(S,u) : u \in \Sigma^*\}\right| < \left|\mathcal{A}\right|$
- $\exists u \quad \{p\} \subseteq \delta(S,u) \subseteq \neg F$

Characterization by set coverage

Set S that covers $p \in \neg F$

- $\left|\{\delta(S,u) : u \in \Sigma^*\}\right| < |\mathcal{A}|$
- $\exists u \quad \{p\} \subseteq \delta(S,u) \subseteq \neg F$

Key equivalence

a permutation DFA is composite

$\Updownarrow$

some sets suffice to cover all non-finals states

Characterization by set coverage

Set S that covers $p \in \neg F$

- $\left|\{\delta(S,u) : u \in \Sigma^*\}\right| < |\mathcal{A}|$
- $\exists u \quad \{p\} \subseteq \delta(S,u) \subseteq \neg F$

Key equivalence

a permutation DFA is composite

$\Updownarrow$

some sets suffice to cover all non-finals states

Deciding compositionality of permutation DFAs (k is not given)

1. For each rejecting state $p \in \neg F$
   - 1.1 Guess S
   - 1.2 Check $\left|\{\delta(S,u) : u \in \Sigma^*\}\right| < |\mathcal{A}|$
   - 1.3 Check $\exists u \quad \{p\} \subseteq \delta(S,u) \subseteq \neg F$

Characterization by set coverage

Set S that covers $p \in \neg F$

- $\left|\{\delta(S, u) : u \in \Sigma^*\}\right| < |\mathcal{A}|$
- $\exists u \quad \{p\} \subseteq \delta(S, u) \subseteq \neg F$

Key equivalence

a permutation DFA is composite

$\Updownarrow$

some sets suffice to cover all non-finals states

Deciding compositionality of permutation DFAs (k is not given)

1. For each rejecting state $p \in \neg F$
   1.1 Guess S
   1.2 Check $\left|\{\delta(S, u) : u \in \Sigma^*\}\right| < |\mathcal{A}|$
   1.3 Check $\exists u \quad \{p\} \subseteq \delta(S, u) \subseteq \neg F$

Theorem

- *Compositionality is in* NP *for permutation DFAs*

Width matters

Minimizing number of factors

Width of $\mathcal{A}$ $=$ minimal k such that $\mathcal{A}$ is k-composite

Minimizing number of factors

Width of $\mathcal{A}$ $\quad = \quad$ minimal k such that $\mathcal{A}$ is k-composite

Decompositions of a natural $n \in \mathbb{N}$

Prime decomposition: $n = n_1 \times n_2 \times \cdots \times n_k$ where all $n_i \in \mathbb{N}$ are prime.

2-Decomposition: $n = n' \times n''$ with $n' < n$ and $n'' < n$.

Minimizing number of factors

Width of $\mathcal{A}$ $=$ minimal k such that $\mathcal{A}$ is k-composite

Decompositions of a natural $n \in \mathbb{N}$

Prime decomposition: $n = n_1 \times n_2 \times \cdots \times n_k$ where all $n_i \in \mathbb{N}$ are prime.

2-Decomposition: $n = n' \times n''$ with $n' < n$ and $n'' < n$.

DFA are set of naturals (alphabet size is the representation base)

- *DFA of width 3 by Kupferman and Mosheiff in [1]*

Minimizing number of factors

Width of $\mathcal{A}$ $=$ minimal k such that $\mathcal{A}$ is k-composite

Decompositions of a natural $n \in \mathbb{N}$

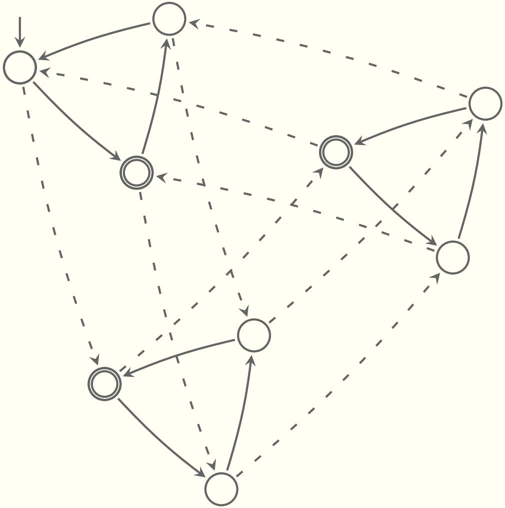
Prime decomposition: $n = n_1 \times n_2 \times \cdots \times n_k$ where all $n_i \in \mathbb{N}$ are prime.

2-Decomposition: $n = n' \times n''$ with $n' < n$ and $n'' < n$.

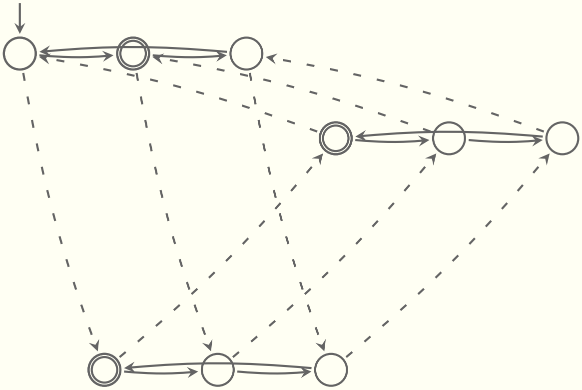
DFA are set of naturals (alphabet size as representation base)

- *DFA of width 3 by Kupferman and Mosheiff in [1]*
- *Family of unary DFAs of unbounded width by Jecker, Mazzocchi and Kupferman in MFCS'20 [2]*

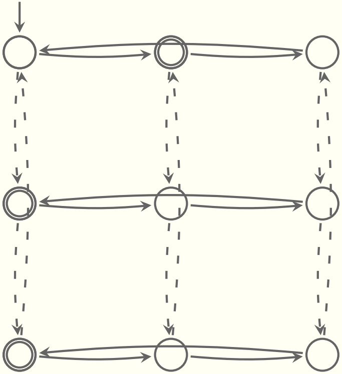
The special case of Torus DFAs



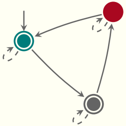
The special case of Torus DFAs



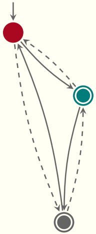
The special case of Torus DFAs



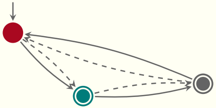
The special case of Torus DFAs



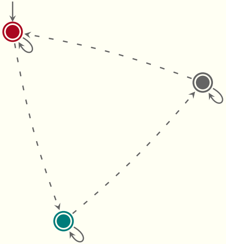
$u = \dashrightarrow$



$u = \dashrightarrow\longrightarrow$

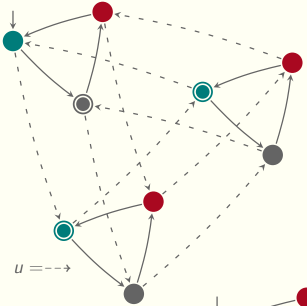


$u = \dashrightarrow\longrightarrow\longrightarrow$

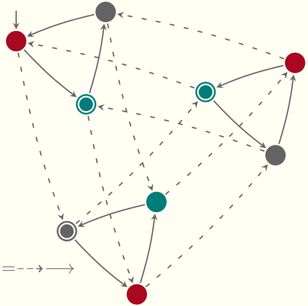


$u = \longrightarrow$

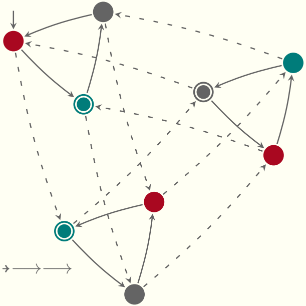
The special case of Torus DFAs



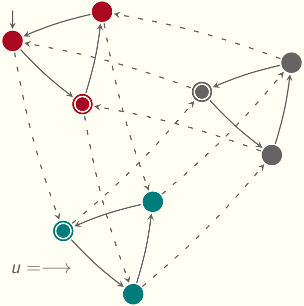
$u = \dashrightarrow$



$u = \dashrightarrow\longrightarrow$

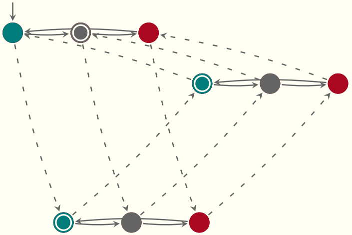


$u = \dashrightarrow\longrightarrow\longrightarrow$

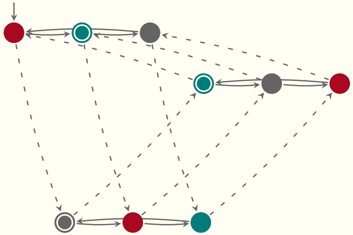


$u = \longrightarrow$

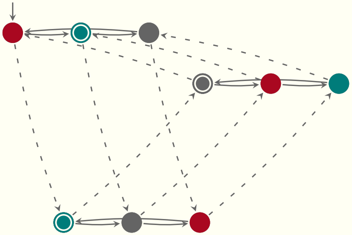
The special case of Torus DFAs



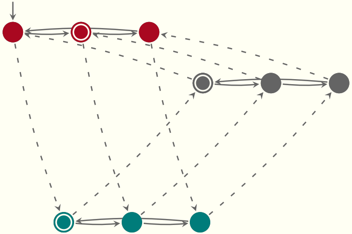
$u = \dashrightarrow$



$u = \dashrightarrow\longrightarrow$

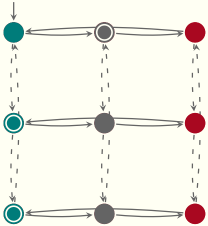


$u = \dashrightarrow\longrightarrow\longrightarrow$

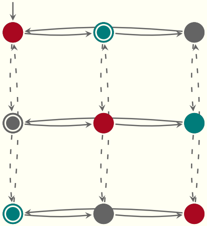


$u = \longrightarrow$

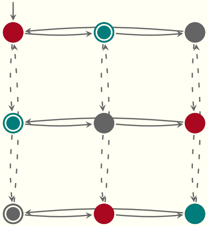
The special case of Torus DFAs



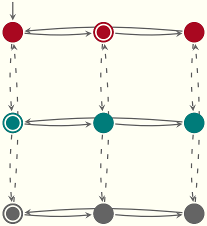
$u = \dashrightarrow$



$u = \dashrightarrow\longrightarrow$

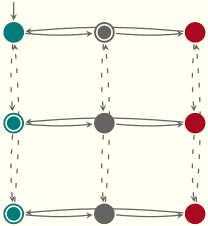


$u = \dashrightarrow\longrightarrow\longrightarrow$

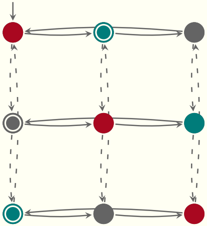


$u = \longrightarrow$

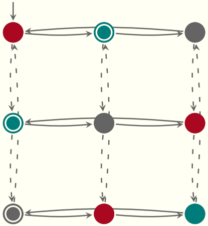
The special case of Torus DFAs



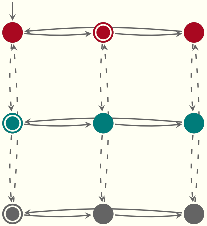
$u = \dashrightarrow$



$u = \dashrightarrow\longrightarrow$



$u = \dashrightarrow\longrightarrow\longrightarrow$



$u = \longrightarrow$

Torus with odd prime size

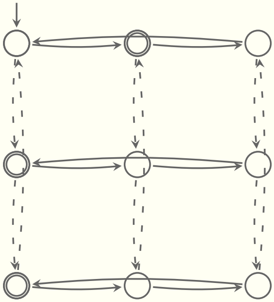
States visited by a word iteration

$\Updownarrow$

Line of the torus

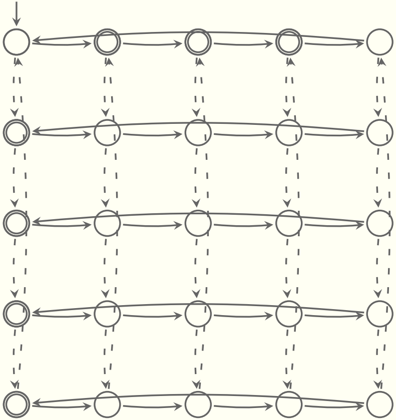
(horizontal, vertical, diagonal i.e. visits **once** every raws & colums)

Toward a large decomposition



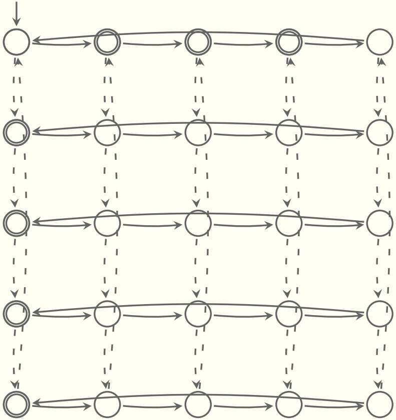
◁· size of dimensions = 3 ·▷

Toward a large decomposition



◁··· size of dimensions $= n$ (odd prime) ···▷

Toward a large decomposition



$\lhd\cdots$ size of dimensions $= n$ (odd prime) $\cdots\rhd$

Compositionality criteron

k-factor composite

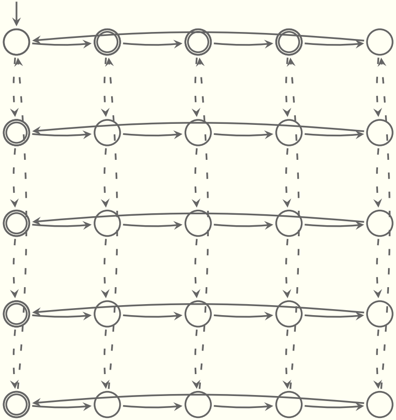
$\Updownarrow$

k words suffice to cover all non-finals

$\Updownarrow$

k lines of non-final suffice to cover them all

Toward a large decomposition



◁··· size of dimensions $= n$ (odd prime) ···▷

Compositionality criteron

k-factor composite

$\Updownarrow$

k words suffice to cover all non-finals

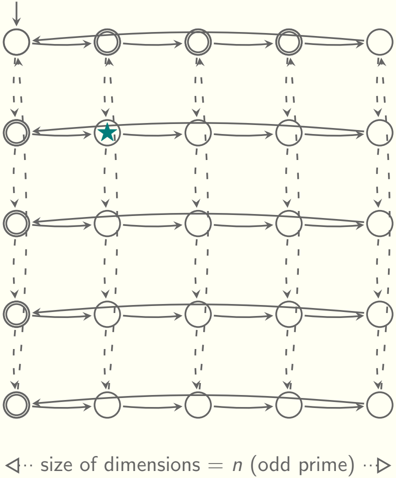
$\Updownarrow$

k lines of non-final suffice to cover them all

Width

minimal number of lines which, together, cover all non-finals

Toward a large decomposition



Compositionality criteron

k-factor composite

$\Updownarrow$

k words suffice to cover all non-finals

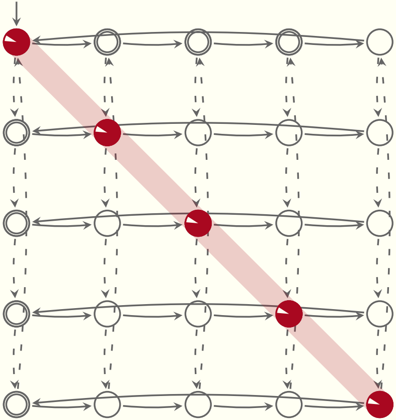
$\Updownarrow$

k lines of non-final suffice to cover them all

Width

minimal number of lines which, together, cover all non-finals

Toward a large decomposition



◁·· size of dimensions $= n$ (odd prime) ··▷

Compositionality criteron

k-factor composite

$\Updownarrow$

k words suffice to cover all non-finals

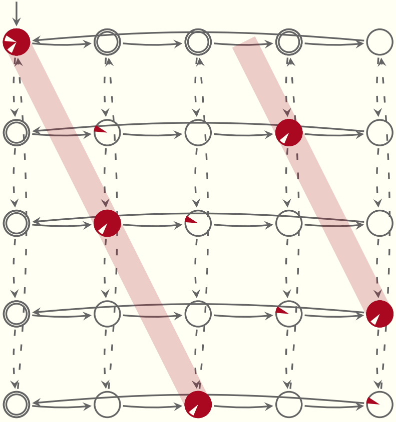
$\Updownarrow$

k lines of non-final suffice to cover them all

Width

minimal number of lines which, together, cover all non-finals

Toward a large decomposition



◁·· size of dimensions $= n$ (odd prime) ··▷

Compositionality criteron

k-factor composite

$\Updownarrow$

k words suffice to cover all non-finals

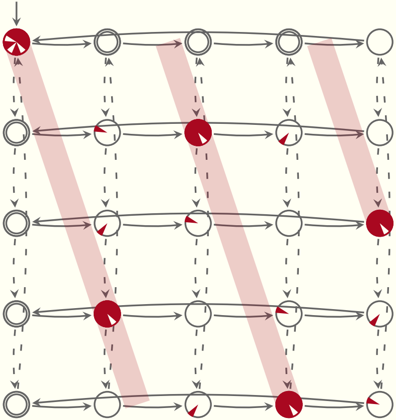
$\Updownarrow$

k lines of non-final suffice to cover them all

Width

minimal number of lines which, together, cover all non-finals

Toward a large decomposition



$\triangleleft \cdots$ size of dimensions $= n$ (odd prime) $\cdots \triangleright$

Compositionality criteron

k-factor composite

$\Updownarrow$

k words suffice to cover all non-finals

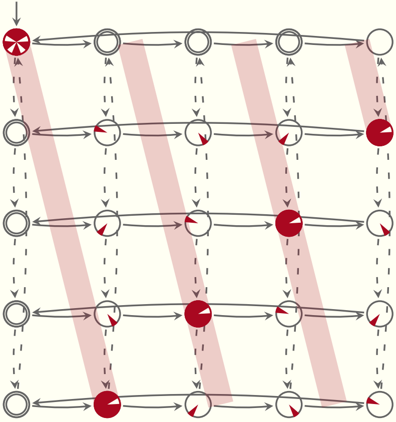
$\Updownarrow$

k lines of non-final suffice to cover them all

Width

minimal number of lines which, together, cover all non-finals

Toward a large decomposition



◁·· size of dimensions = n (odd prime) ··▷

Compositionality criteron

k-factor composite

⇕

k words suffice to cover all non-finals

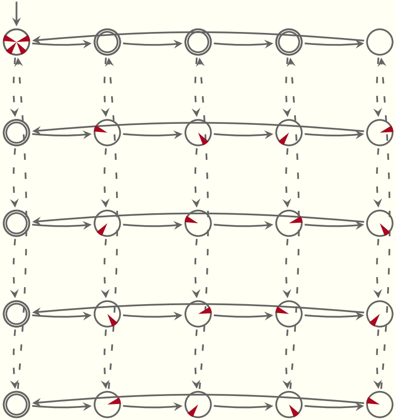
⇕

k lines of non-final suffice to cover them all

Width

minimal number of lines which, together, cover all non-finals

Toward a large decomposition



Compositionality criteron

k-factor composite

$\Updownarrow$

k words suffice to cover all non-finals

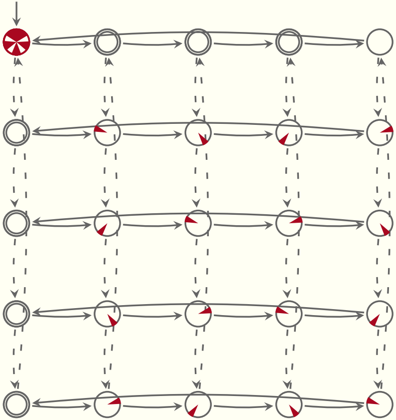
$\Updownarrow$

k lines of non-final suffice to cover them all

Width

minimal number of lines which, together, cover all non-finals

Toward a large decomposition



◁·· size of dimensions $= n$ (odd prime) ··▷

Compositionality criteron

k-factor composite

$\Updownarrow$

k words suffice to cover all non-finals

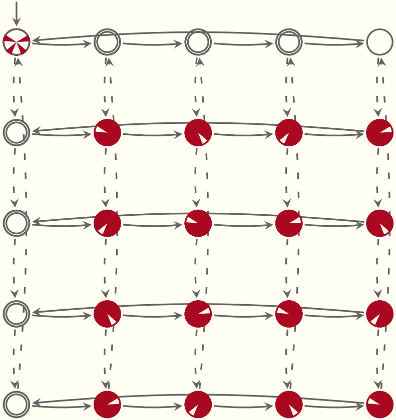
$\Updownarrow$

k lines of non-final suffice to cover them all

Width

minimal number of lines which, together, cover all non-finals

Toward a large decomposition



◁·· size of dimensions $= n$ (odd prime) ··▷

Compositionality criteron

k-factor composite

$\Updownarrow$

k words suffice to cover all non-finals

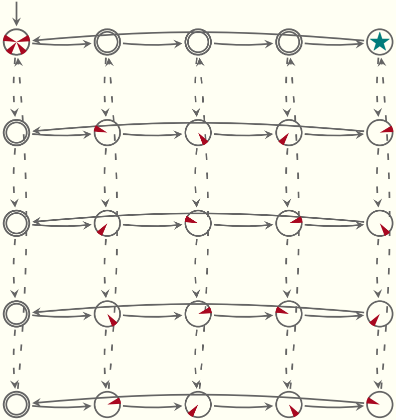
$\Updownarrow$

k lines of non-final suffice to cover them all

Width

minimal number of lines which, together, cover all non-finals

Toward a large decomposition



◁·· size of dimensions $= n$ (odd prime) ··▷

Compositionality criteron

k-factor composite

⇕

k words suffice to cover all non-finals

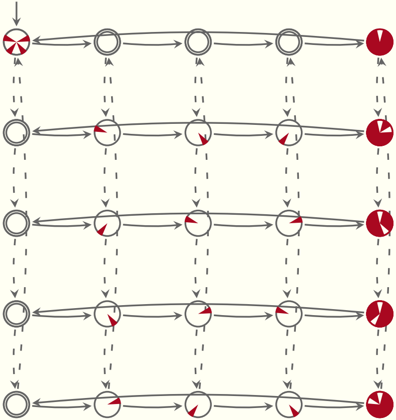
⇕

k lines of non-final suffice to cover them all

Width

minimal number of lines which, together, cover all non-finals

Toward a large decomposition



$\triangleleft\cdots$ size of dimensions $= n$ (odd prime) $\cdots\triangleright$

Compositionality criteron

k-factor composite

$\Updownarrow$

k words suffice to cover all non-finals

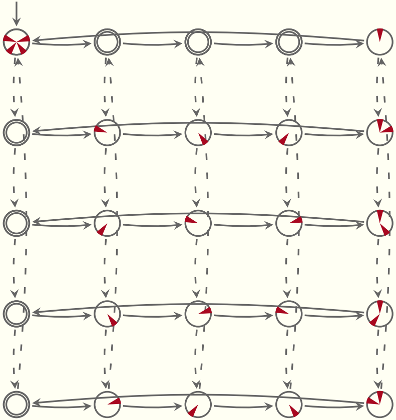
$\Updownarrow$

k lines of non-final suffice to cover them all

Width

minimal number of lines which, together, cover all non-finals

Toward a large decomposition



◁·· size of dimensions $= n$ (odd prime) ··▷

Compositionality criteron

k-factor composite

$\Updownarrow$

k words suffice to cover all non-finals

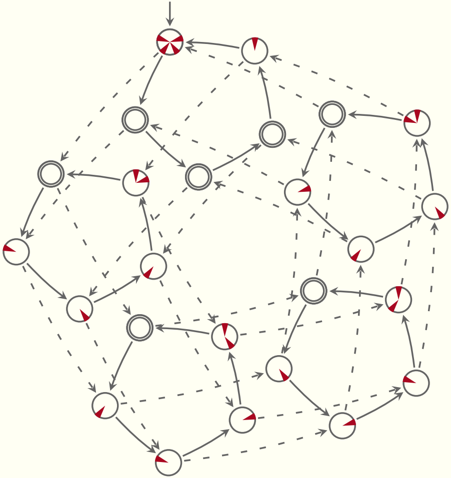
$\Updownarrow$

k lines of non-final suffice to cover them all

Width

minimal number of lines which, together, cover all non-finals

Toward a large decomposition



Compositionality criteron

k-factor composite
$\Updownarrow$
k words suffice to cover all non-finals
$\Updownarrow$
k lines of non-final suffice to cover them all

Width

minimal number of lines which, together, cover all non-finals

Theorem

The Torus DFA with odd prime size n requires $\sqrt{n}$ factors to be decomposed

In a nutshell

	composite?	k-composite?
DFAs	ExpSpace [1]	PSpace
permutation DFAs	~~PSpace [1]~~ NP/FPT	
commutative permutation DFAs	~~PTime [1]~~ NLogSpace	NP-complete
unary (i.e. singleton alphabet) DFAs	LogSpace [2]	LogSpace

1 O. Kupferman and J. Mosheiff
Prime Languages
In *J. of Information and Computation* 240, 2015

2 I. Jecker, O. Kupferman and N. Mazzocchi
Unary Prime Languages
In *MFCS* proceedings, 2020

In a nutshell

	composite?	k-composite?
DFAs	EXPSPACE [1]	PSPACE
permutation DFAs	~~PSPACE [1]~~ NP/FPT	
commutative permutation DFAs	~~PTIME [1]~~ NLOGSPACE	NP-complete
unary (i.e. singleton alphabet) DFAs	LOGSPACE [2]	LOGSPACE

Large decompositions (n states and m letters)

- unary commutative permutation DFAs requiring $\ln(n)/\ln(\ln(n))$ factors to be decomposed [2]
- commutative permutation DFAs requiring $(\sqrt[m]{n}-1)^{m-1}$ factors to be decomposed

1. O. Kupferman and J. Mosheiff
Prime Languages
In *J. of Information and Computation* 240, 2015

2. I. Jecker, O. Kupferman and N. Mazzocchi
Unary Prime Languages
In *MFCS* proceedings, 2020

appendix

Sketch: Characterization by coverage

back

k-covered implies k-composite

1. $\forall p \in \neg F$ u_p covers p
2. $p_1 \xrightarrow{a} p_2 \Leftrightarrow \delta(p_1, u_p^i) \xrightarrow{a} \delta(p_2, u_p^i)$

$$
\begin{aligned}
w \notin L(\mathcal{A}) \quad & \Longleftrightarrow \exists p \in \neg F,\ \mathcal{A}\colon p_I \xrightarrow{w} p \\
& \overset{2}{\Longleftrightarrow} \exists p \in \neg F,\ \mathcal{B}_p\colon [p_I] \xrightarrow{w} [p] \\
& \overset{1}{\Longleftrightarrow} \exists p \in \neg F,\ w \notin L(\mathcal{B}_p) \\
& \Longleftrightarrow w \notin \textstyle\bigcap_p L(\mathcal{B}_p)
\end{aligned}
$$

- $L(\mathcal{A}) = \bigcap_p L(\mathcal{B}_p)$

Sketch: Characterization by coverage

⏩ back

k-covered implies k-composite

1. $\forall p \in \neg F$ u_p covers p
2. $\boxed{p_1 \xrightarrow{a} p_2 \Leftrightarrow \delta(p_1, u_p^i) \xrightarrow{a} \delta(p_2, u_p^i)}$

$$
\begin{aligned}
w \notin L(\mathcal{A}) &\iff \exists p \in \neg F,\ \mathcal{A} \colon p_I \xrightarrow{w} p \\
&\overset{2}{\iff} \exists p \in \neg F,\ \mathcal{B}_p \colon [p_I] \xrightarrow{w} [p] \\
&\overset{1}{\iff} \exists p \in \neg F,\ w \notin L(\mathcal{B}_p) \\
&\iff w \notin \bigcap_p L(\mathcal{B}_p)
\end{aligned}
$$

- $L(\mathcal{A}) = \bigcap_p L(\mathcal{B}_p)$

k-composite implies k-covered

1. $L(\mathcal{A}) = \bigcap_i L(\mathcal{B}_i)$
2. $\forall \mathcal{B}_i \begin{cases} w \notin L(\mathcal{B}_i) \implies w \notin L(\mathcal{A}) \\ |\mathcal{B}_i| < |\mathcal{A}| \end{cases}$
3. $\boxed{\exists \lambda\ \delta(p, u^\lambda) = p}$

$$\exists v_i \begin{cases} \mathcal{B}_i \colon q \xrightarrow{v_i} q \\ \mathcal{A} \colon p \xrightarrow{v_i} p' \neq p \end{cases}$$

$$
\begin{aligned}
w \notin L(\mathcal{A}) &\overset{1}{\implies} \exists i,\ w \notin L(\mathcal{B}_i) \\
&\overset{3}{\implies} \exists i,\ wv_i^* \notin L(\mathcal{B}_i) \\
&\overset{2}{\implies} \exists i,\ wv_i^* \notin L(\mathcal{A})
\end{aligned}
$$

- All v_i cover together rejecting states of $\mathcal{A}$

Sketch: Characterization by coverage

back

k-covered implies k-composite

1. $\forall p \in \neg F$ u_p covers p
2. $\boxed{p_1 \xrightarrow{a} p_2 \Leftrightarrow \delta(p_1, u_p^i) \xrightarrow{a} \delta(p_2, u_p^i)}$

$$
\begin{aligned}
w \notin L(\mathcal{A}) &\iff \exists p \in \neg F,\ \mathcal{A}\colon p_I \xrightarrow{w} p \\
&\overset{2}{\iff} \exists p \in \neg F,\ \mathcal{B}_p\colon [p_I] \xrightarrow{w} [p] \\
&\overset{1}{\iff} \exists p \in \neg F,\ w \notin L(\mathcal{B}_p) \\
&\iff w \notin \textstyle\bigcap_p L(\mathcal{B}_p)
\end{aligned}
$$

- $L(\mathcal{A}) = \bigcap_p L(\mathcal{B}_p)$

k-composite implies k-covered

1. $L(\mathcal{A}) = \bigcap_i L(\mathcal{B}_i)$
2. $\forall \mathcal{B}_i \begin{cases} w \notin L(\mathcal{B}_i) \implies w \notin L(\mathcal{A}) \\ |\mathcal{B}_i| < |\mathcal{A}| \end{cases}$
3. $\boxed{\exists \lambda\ \delta(p, u^\lambda) = p}$

$$
\exists v_i \begin{cases} \mathcal{B}_i\colon q \xrightarrow{v_i} q \\ \mathcal{A}\colon p \xrightarrow{v_i} p' \neq p \end{cases}
$$

$$
\begin{aligned}
w \notin L(\mathcal{A}) &\overset{1}{\implies} \exists i,\ w \notin L(\mathcal{B}_i) \\
&\overset{3}{\implies} \exists i,\ wv_i^* \notin L(\mathcal{B}_i) \\
w \notin L(\mathcal{A}) &\overset{2}{\implies} \exists i,\ wv_i^* \notin L(\mathcal{A})
\end{aligned}
$$

- All v_i cover together rejecting states of $\mathcal{A}$