RuFiDiM

Petrozavodsk, September 15th, 2014

Approximation of reset thresholds with greedy algorithms

Dmitry Ananichev, Vladimir Gusev

Institute of Mathematics and Computer Science,

Ural Federal University, Ekaterinburg, Russia

Definitions

A DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ $(n = |Q|)$

Definitions

A DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ $(n = |Q|)$

The transition function δ extends in a unique way to $\delta : Q \times \Sigma^* \to Q$.

Definitions

A DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ ($n = |Q|$)

The transition function δ extends in a unique way to $\delta : Q \times \Sigma^* \to Q$.

We use notation $S.w$ for $\{q \in Q | \exists p \in S,\ \delta(p, w) = q\}$, where $S \subseteq Q$.

Definitions

A DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ $(n = |Q|)$

The transition function δ extends in a unique way to $\delta : Q \times \Sigma^* \to Q$.

We use notation $S.w$ for $\{q \in Q | \exists p \in S,\ \delta(p, w) = q\}$, where $S \subseteq Q$.

The word w is a *reset* word for $\mathscr{A}$ if $|Q.w| = 1$.

Definitions

A DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ ($n = |Q|$)

The transition function δ extends in a unique way to $\delta : Q \times \Sigma^* \to Q$.

We use notation $S.w$ for $\{q \in Q | \exists p \in S,\ \delta(p, w) = q\}$, where $S \subseteq Q$.

The word w is a *reset* word for $\mathscr{A}$ if $|Q.w| = 1$.

DFA $\mathscr{A}$ is *synchronizing* if there is a reset word for $\mathscr{A}$

Definitions

A DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ ($n = |Q|$)

The transition function δ extends in a unique way to $\delta : Q \times \Sigma^* \to Q$.

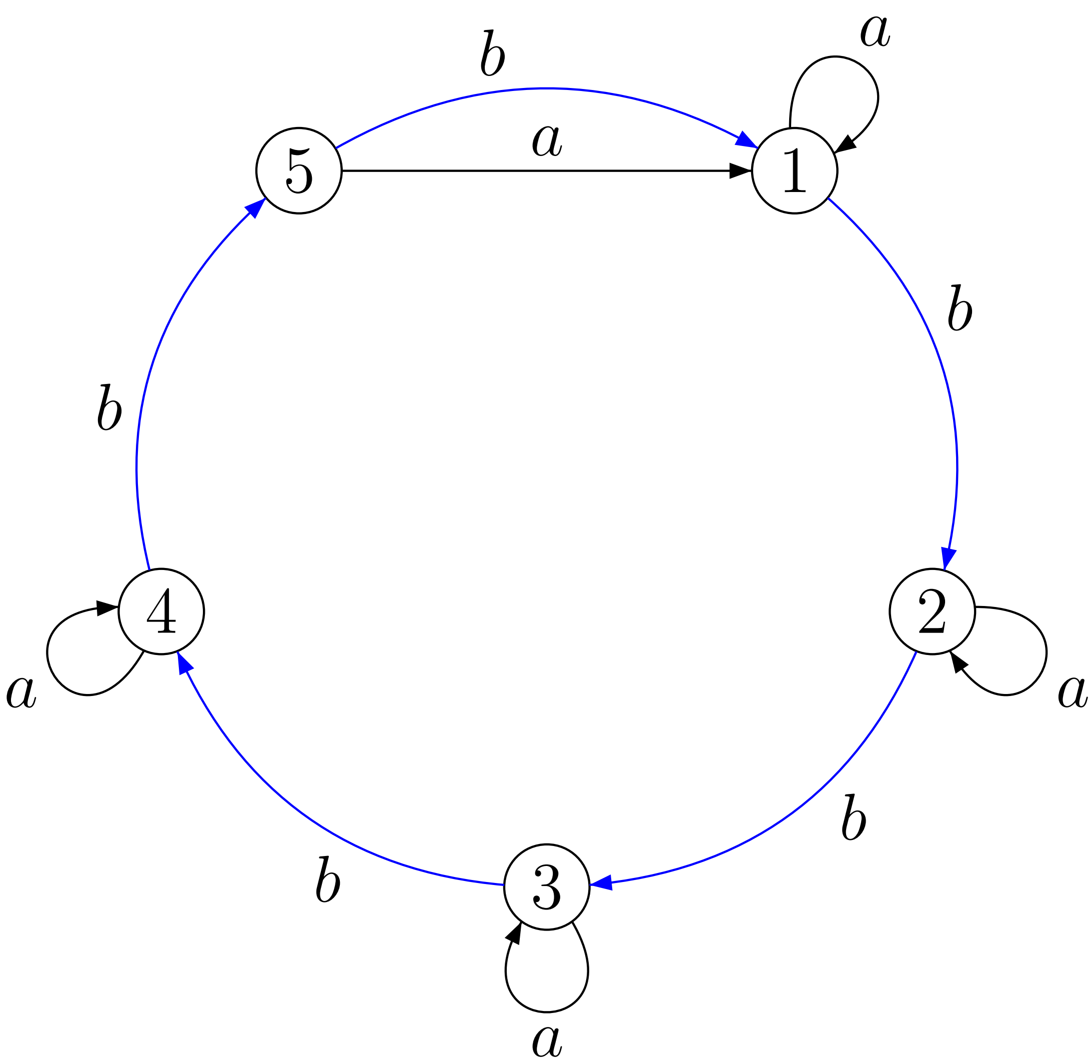
We use notation $S.w$ for $\{q \in Q | \exists p \in S,\ \delta(p, w) = q\}$, where $S \subseteq Q$.

The word w is a *reset* word for $\mathscr{A}$ if $|Q.w| = 1$.

DFA $\mathscr{A}$ is *synchronizing* if there is a reset word for $\mathscr{A}$

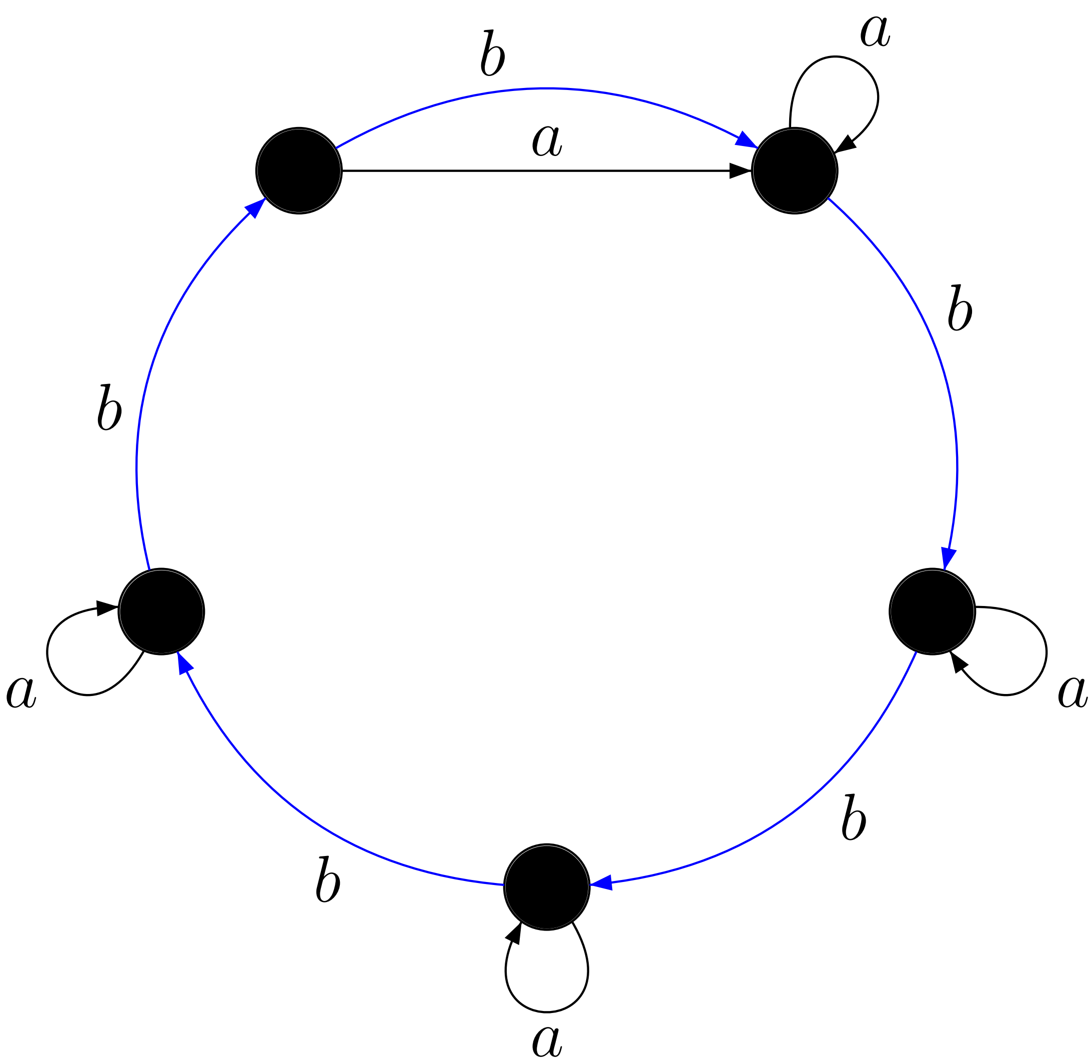
The length of the shortest synchronizing word is called *reset threshold* of $\mathscr{A}$.

reset word



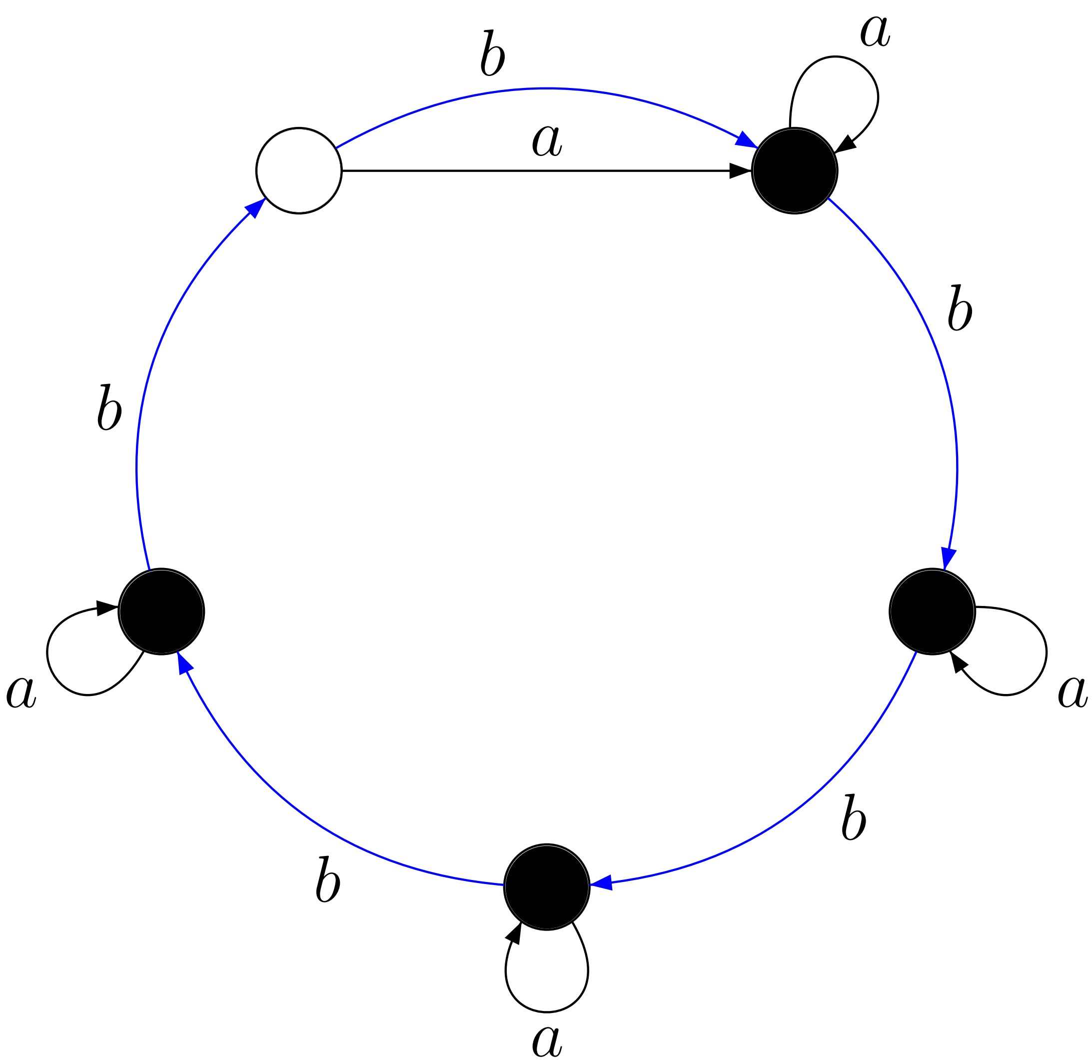
$abbbbabbbbabbbba$

reset word



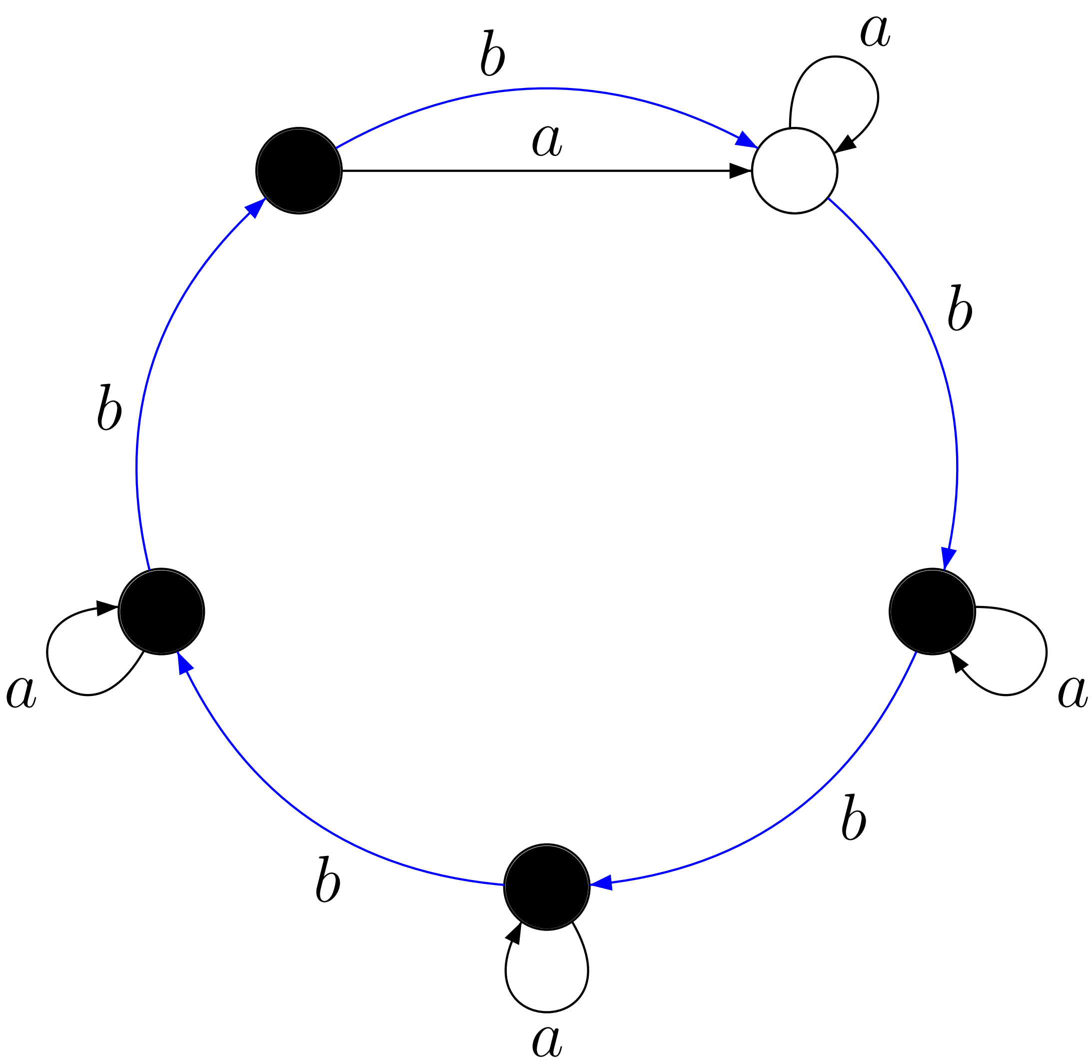
$abbbbabbbbabbbba$

reset word



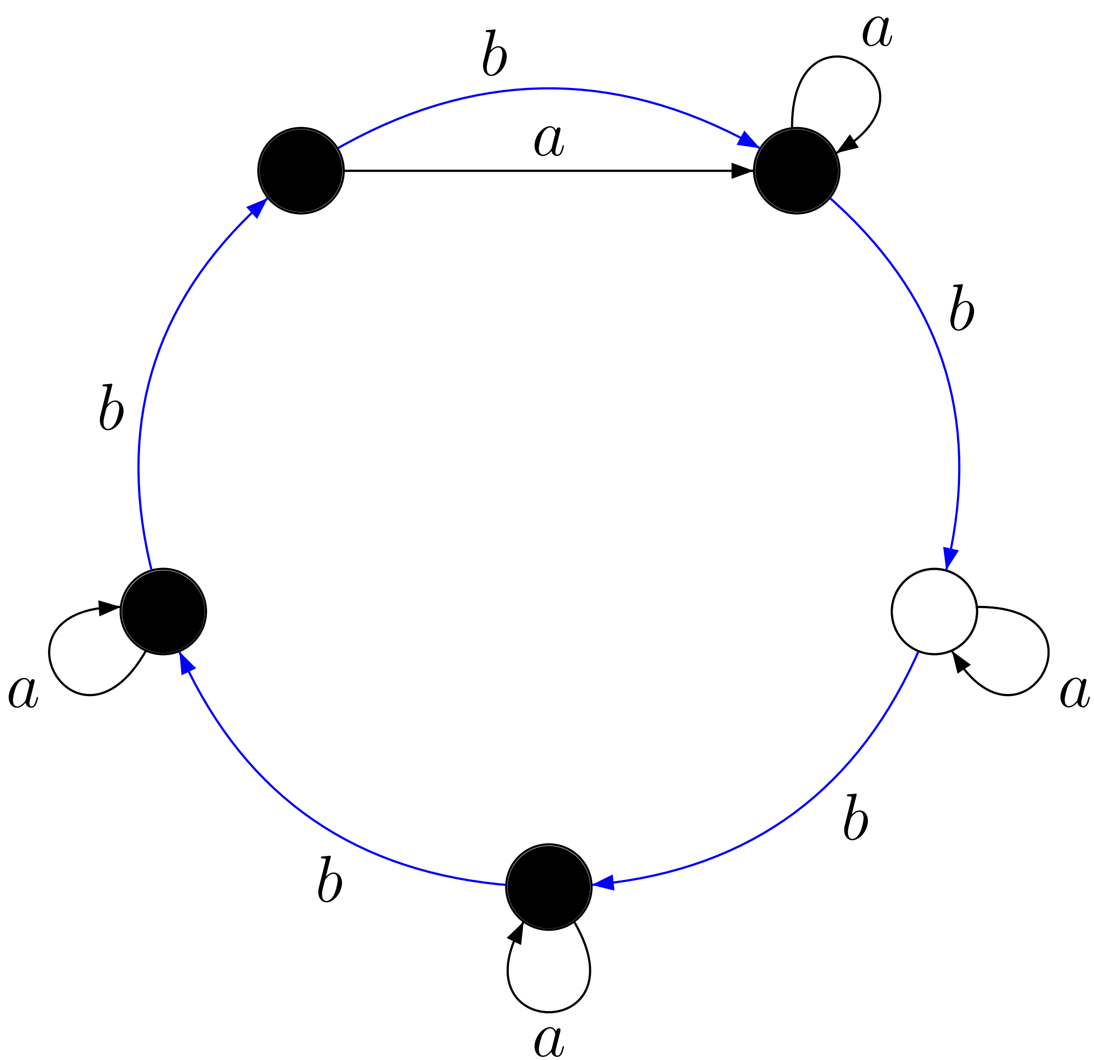
$\color{red}{a}bbbbabbbbabbbba$

reset word



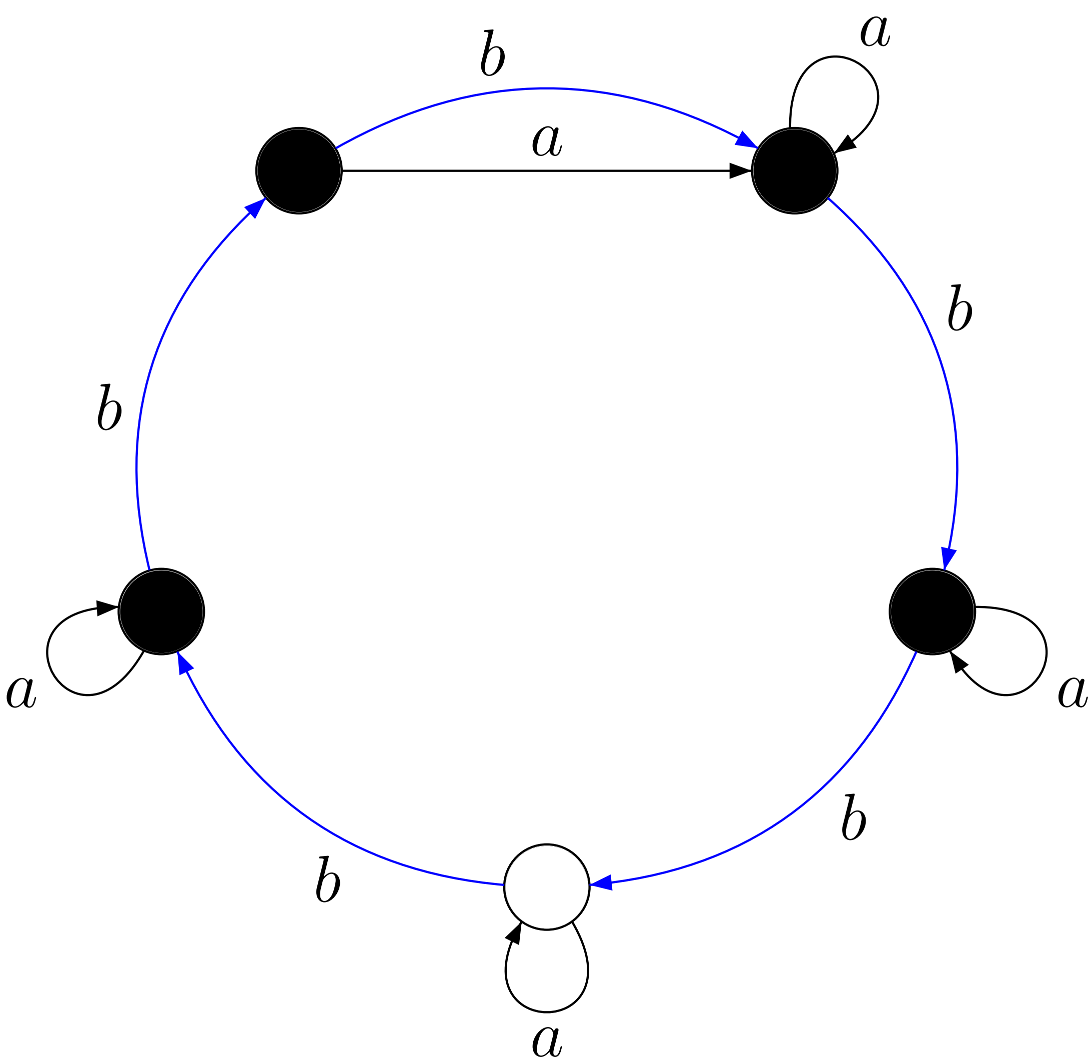
$abbbbabbbbabbbba$

reset word



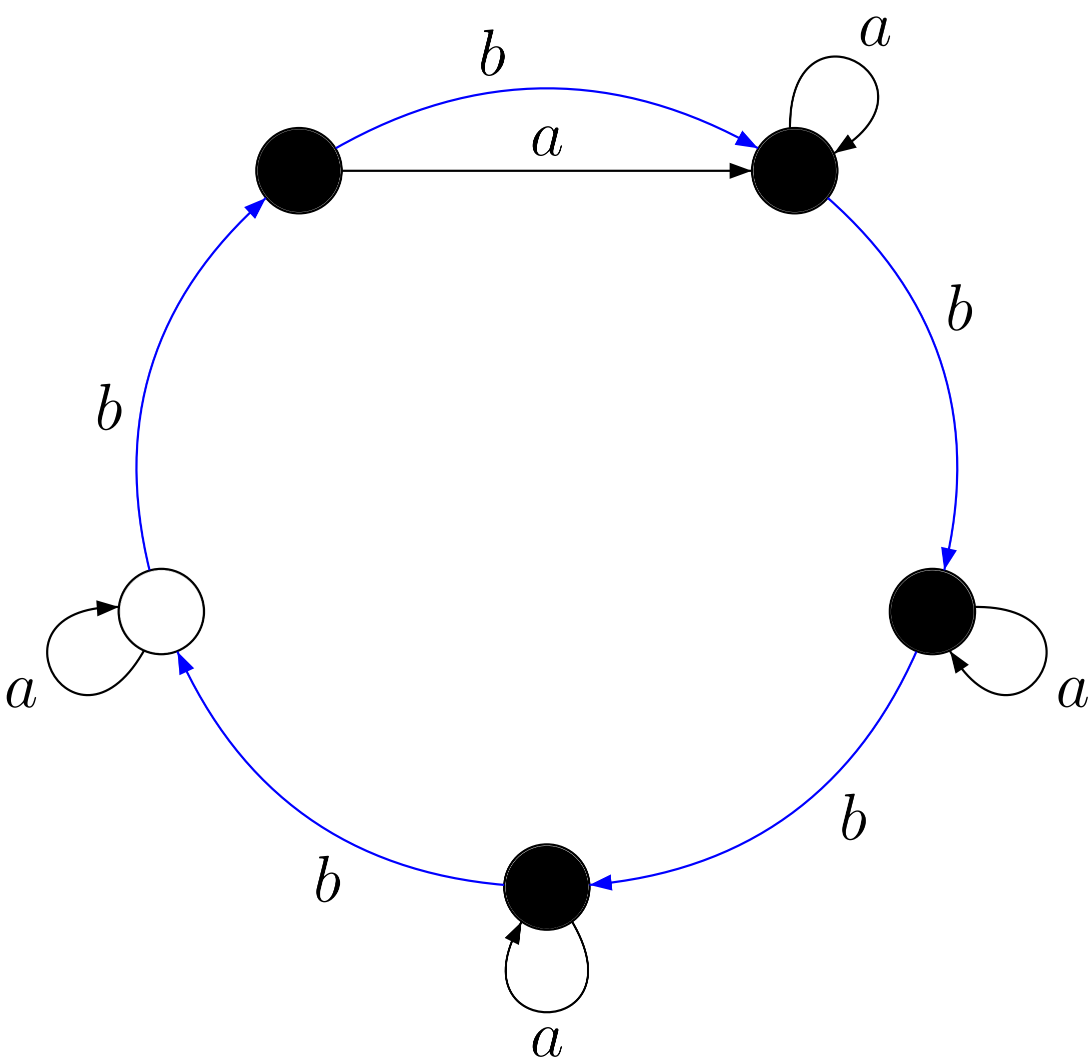
$abbbbabbbbabbbba$

reset word



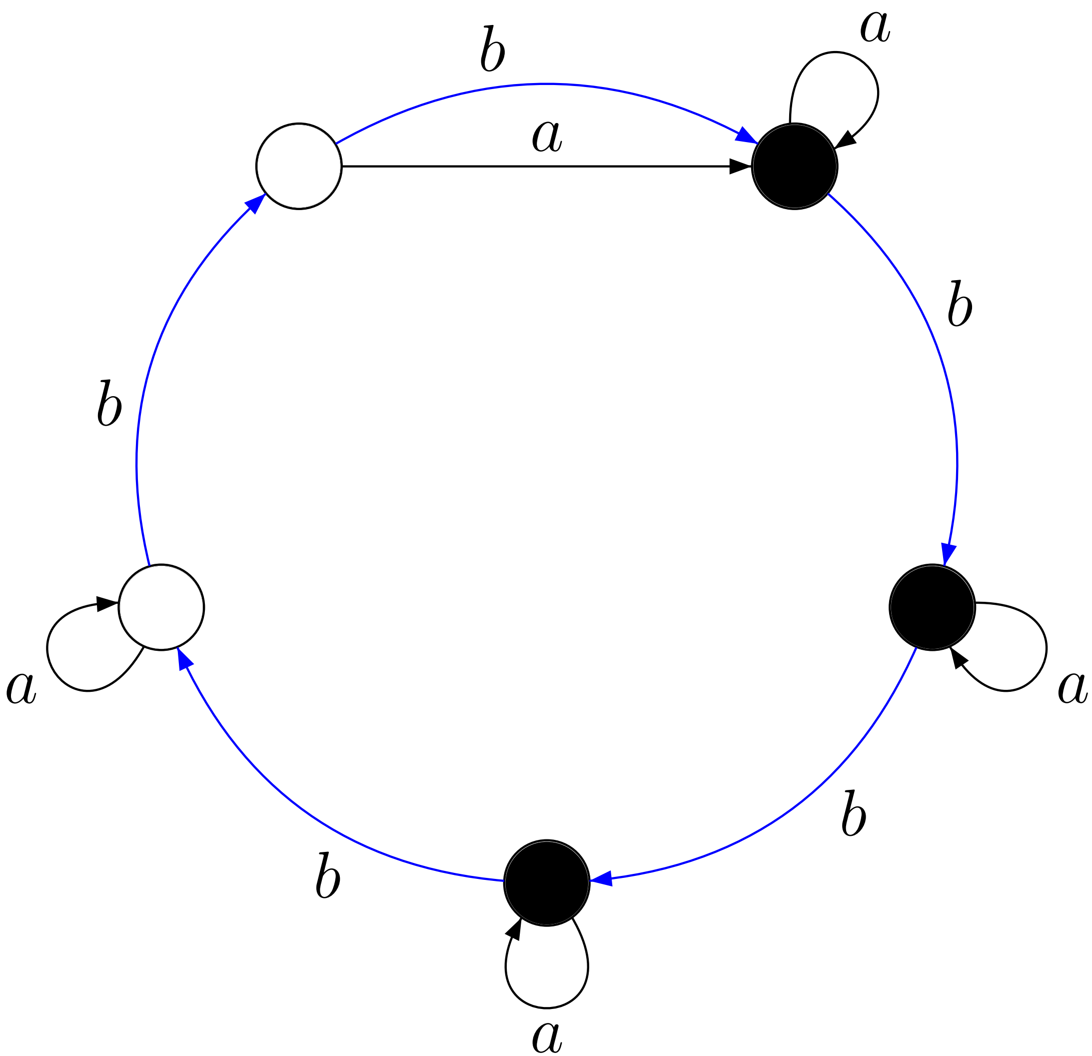
$abb\color{red}{b}\color{black}{babbbbabbbba}$

reset word



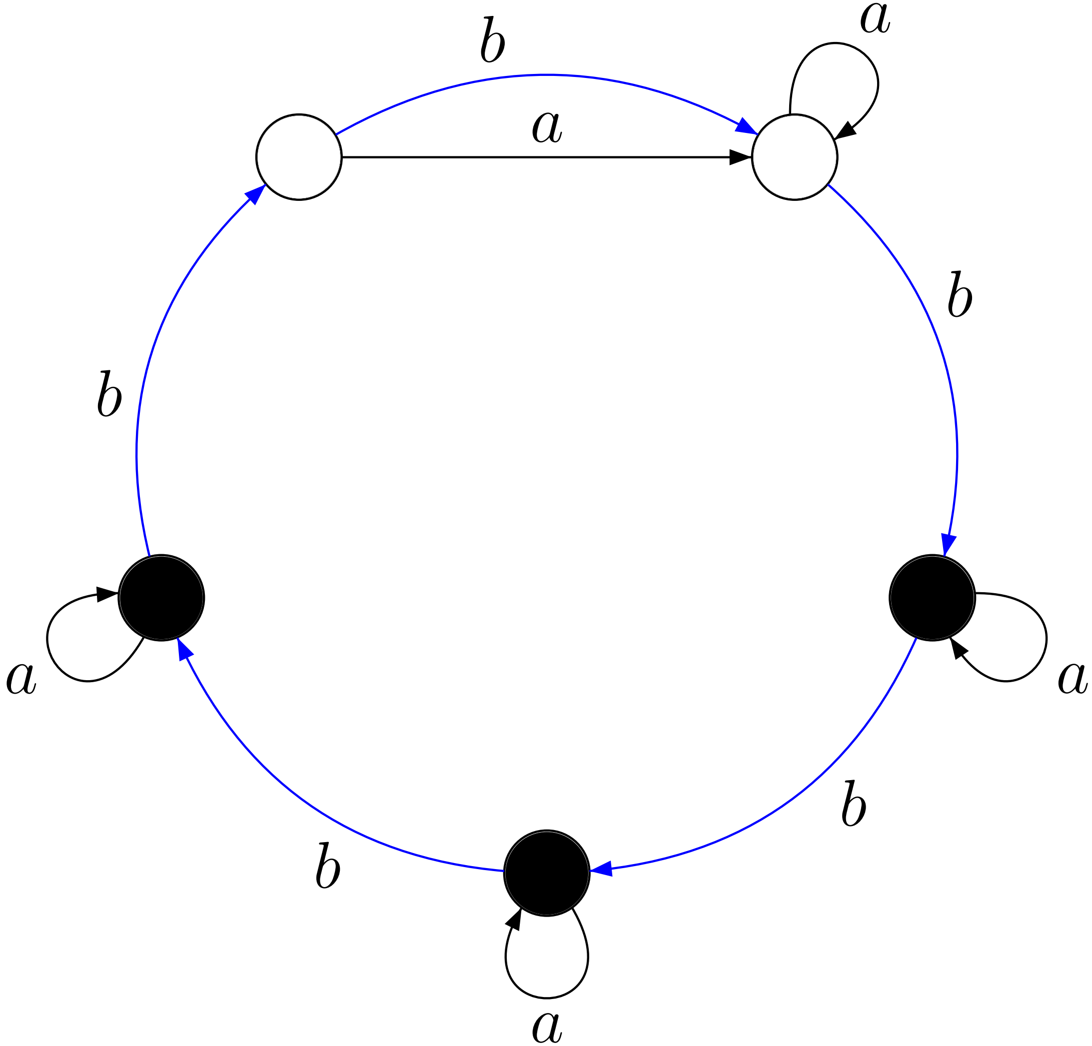
$abbbbabbbbabbbba$

reset word



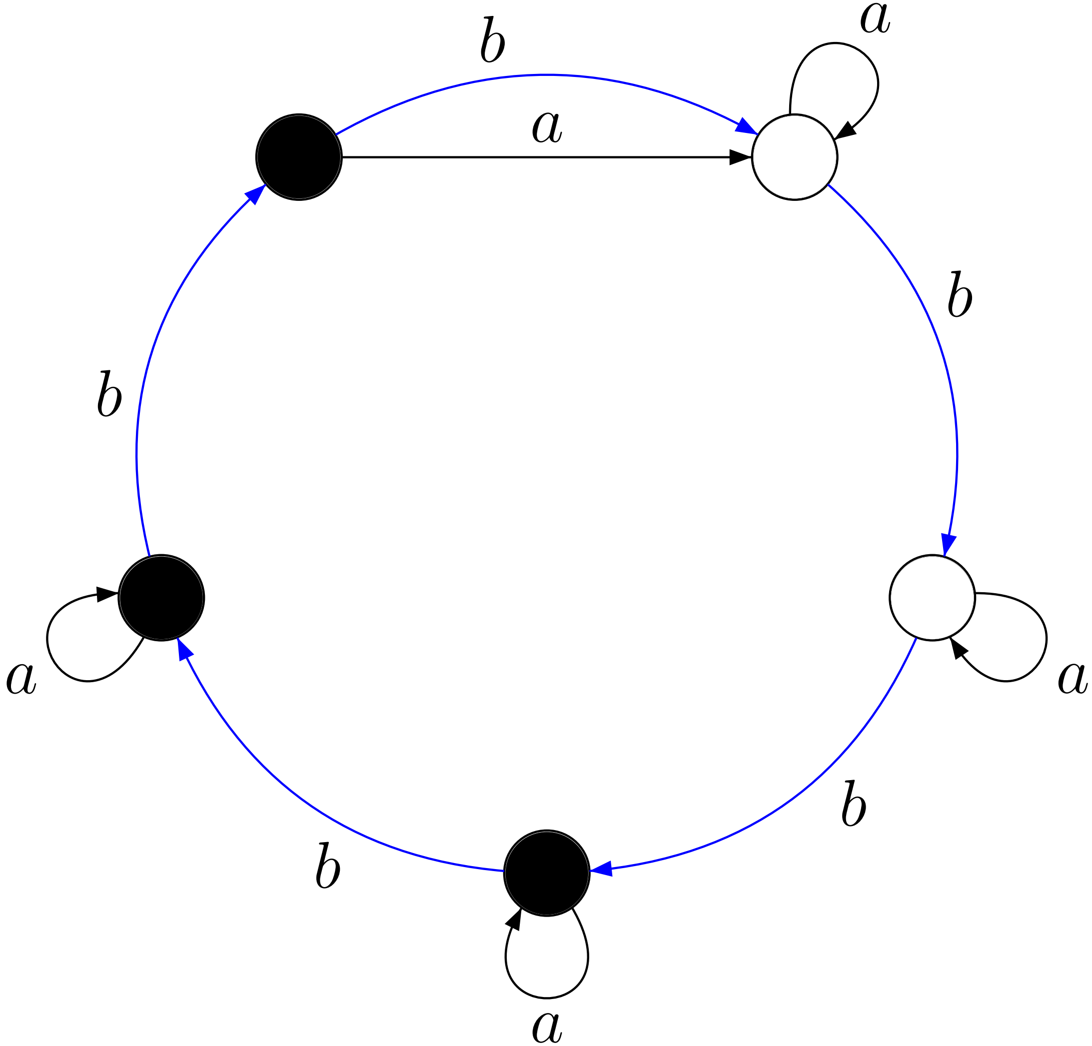
$ab bbbabbbbabbbba$

reset word



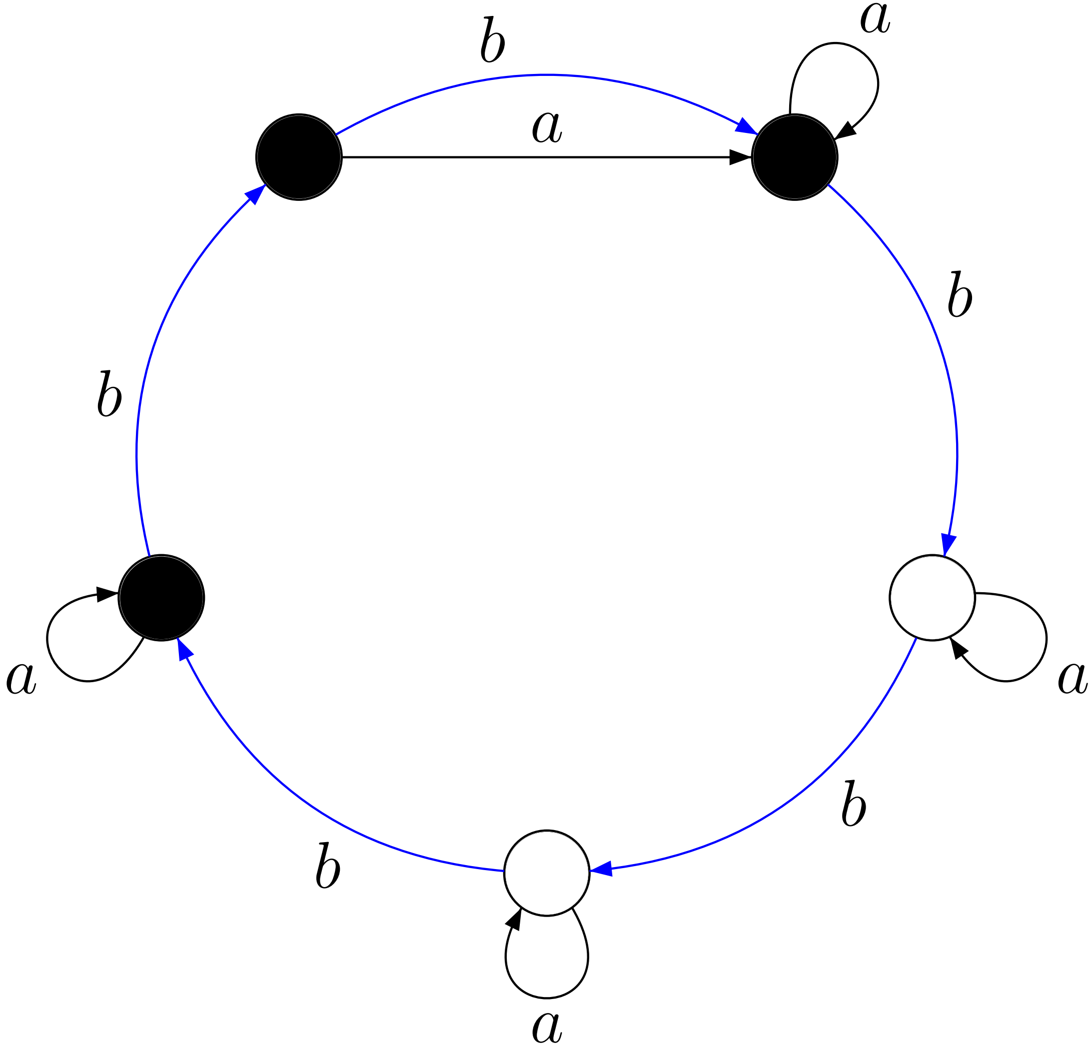
$abbbbabbbbabbbba$

reset word



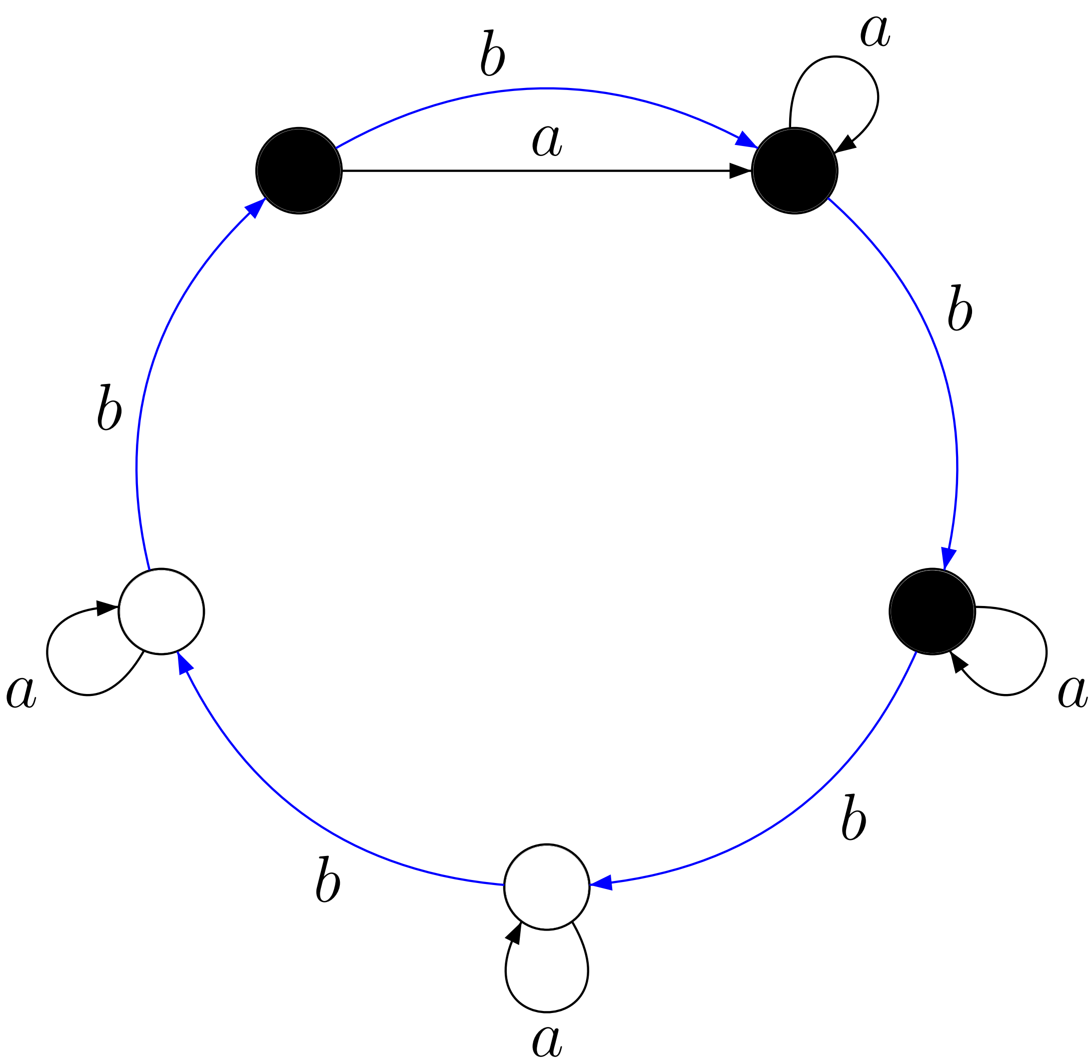
$abbbbabbbbabbbba$

reset word



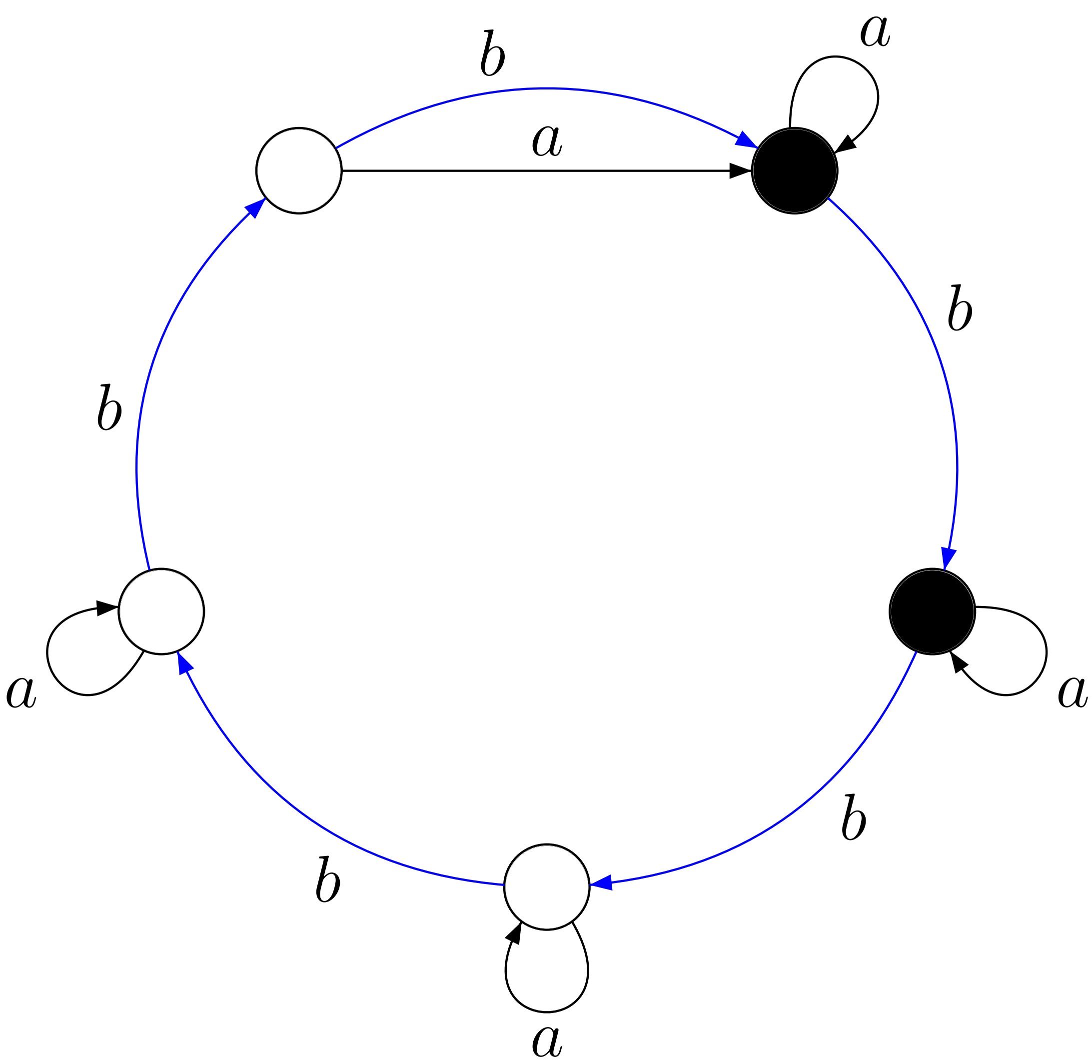
$abbbbabbbbabbbba$

reset word



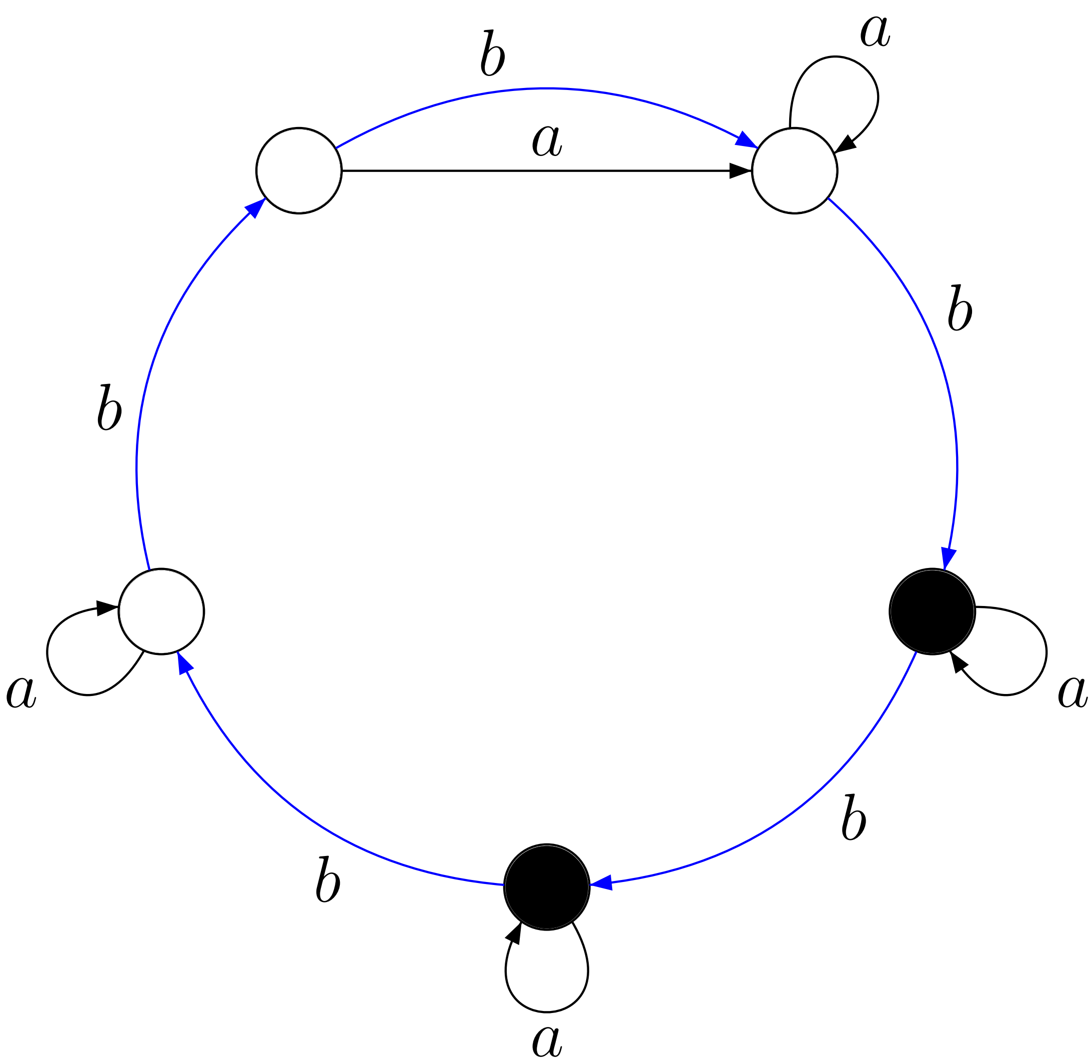
$abbbbabbb{\color{red}b}abbbba$

reset word



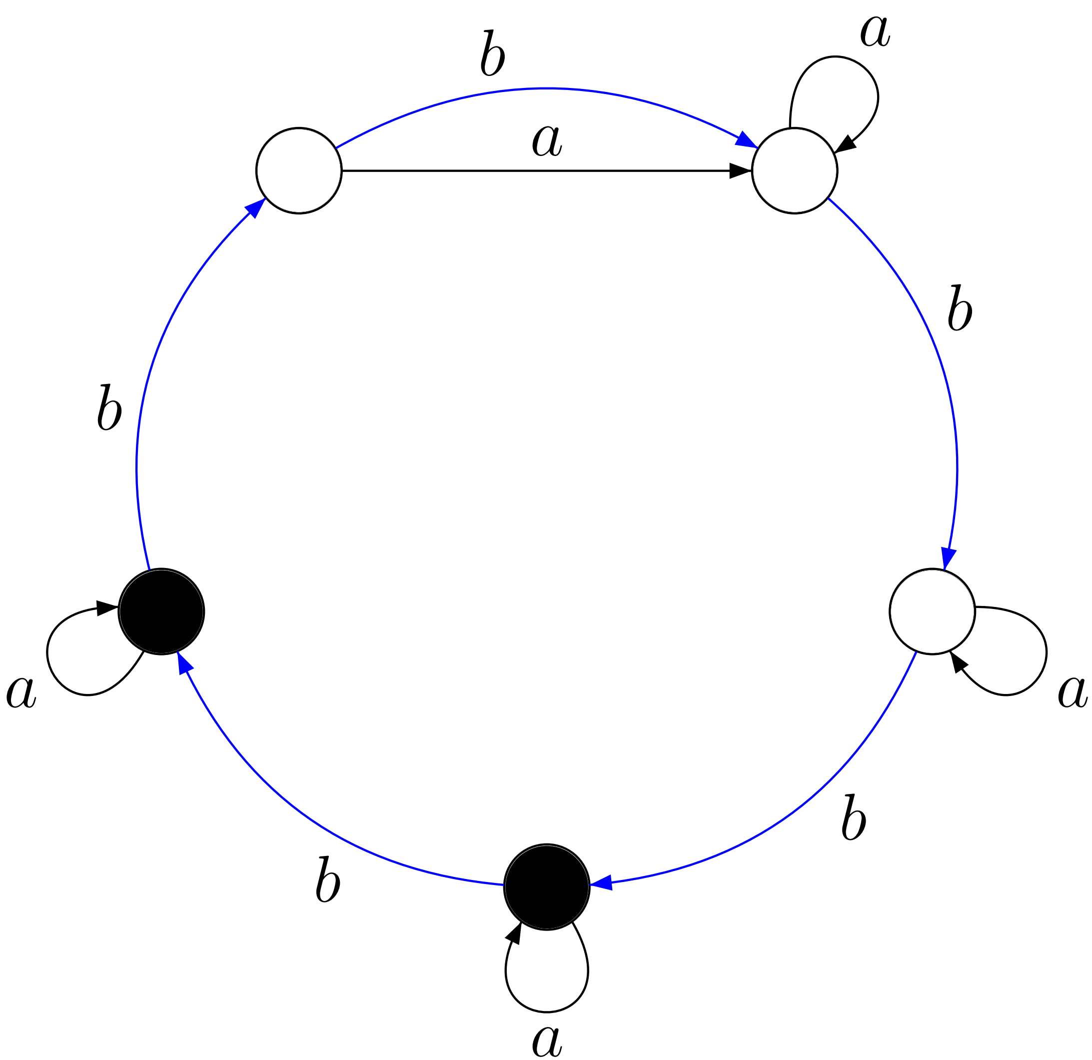
$abbbbabbbbabbbba$

reset word



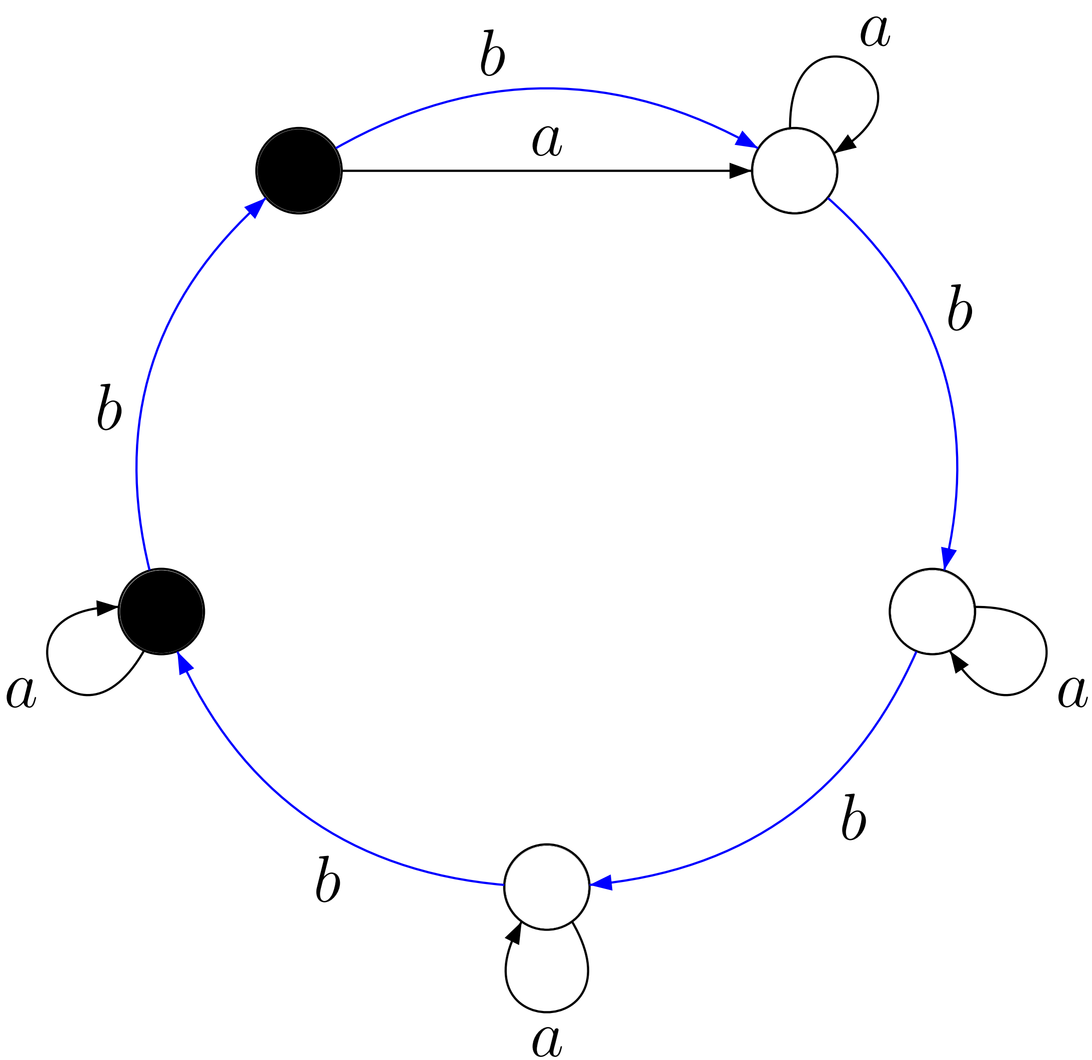
$abbbbabbbbabbbba$

reset word



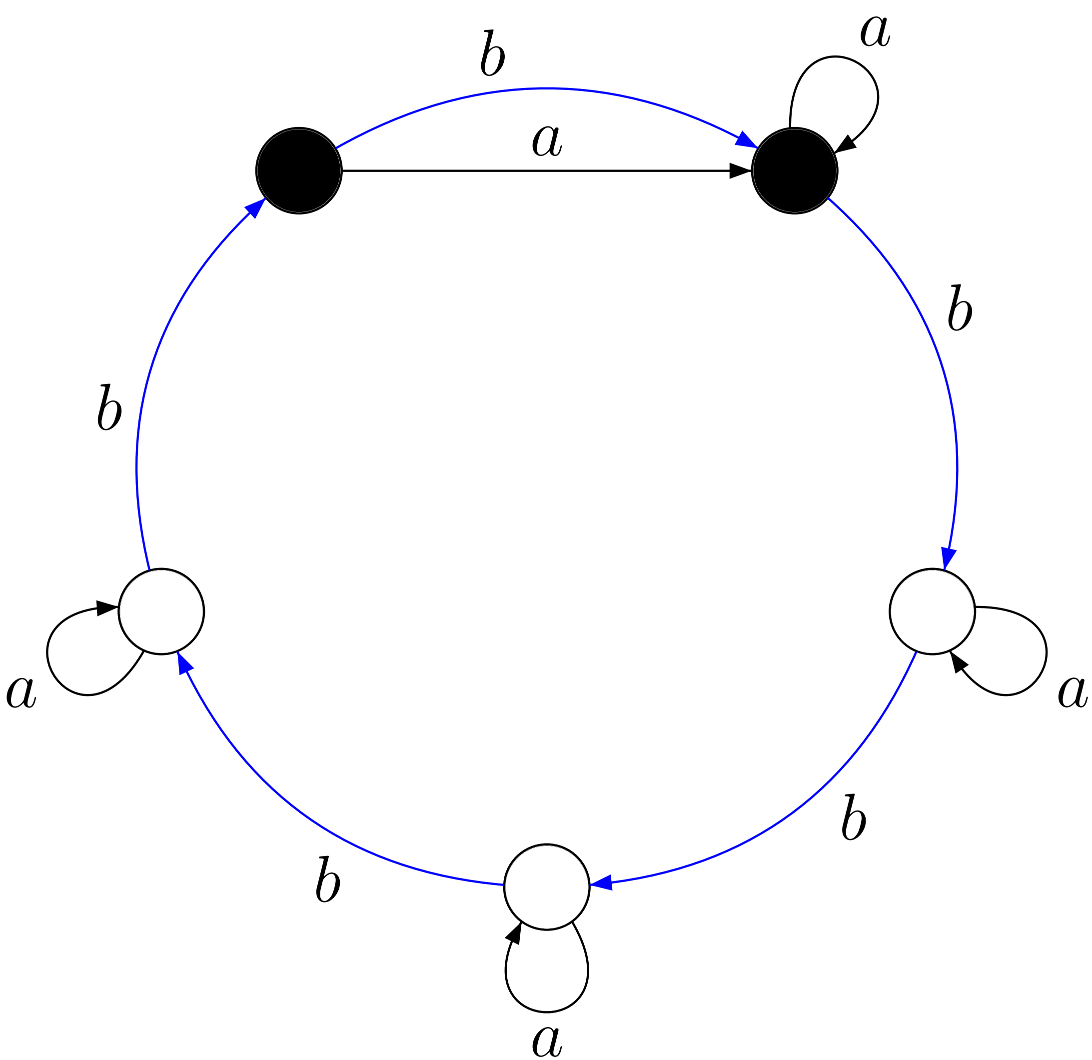
$abbbbabbbbab\textcolor{red}{b}bba$

reset word



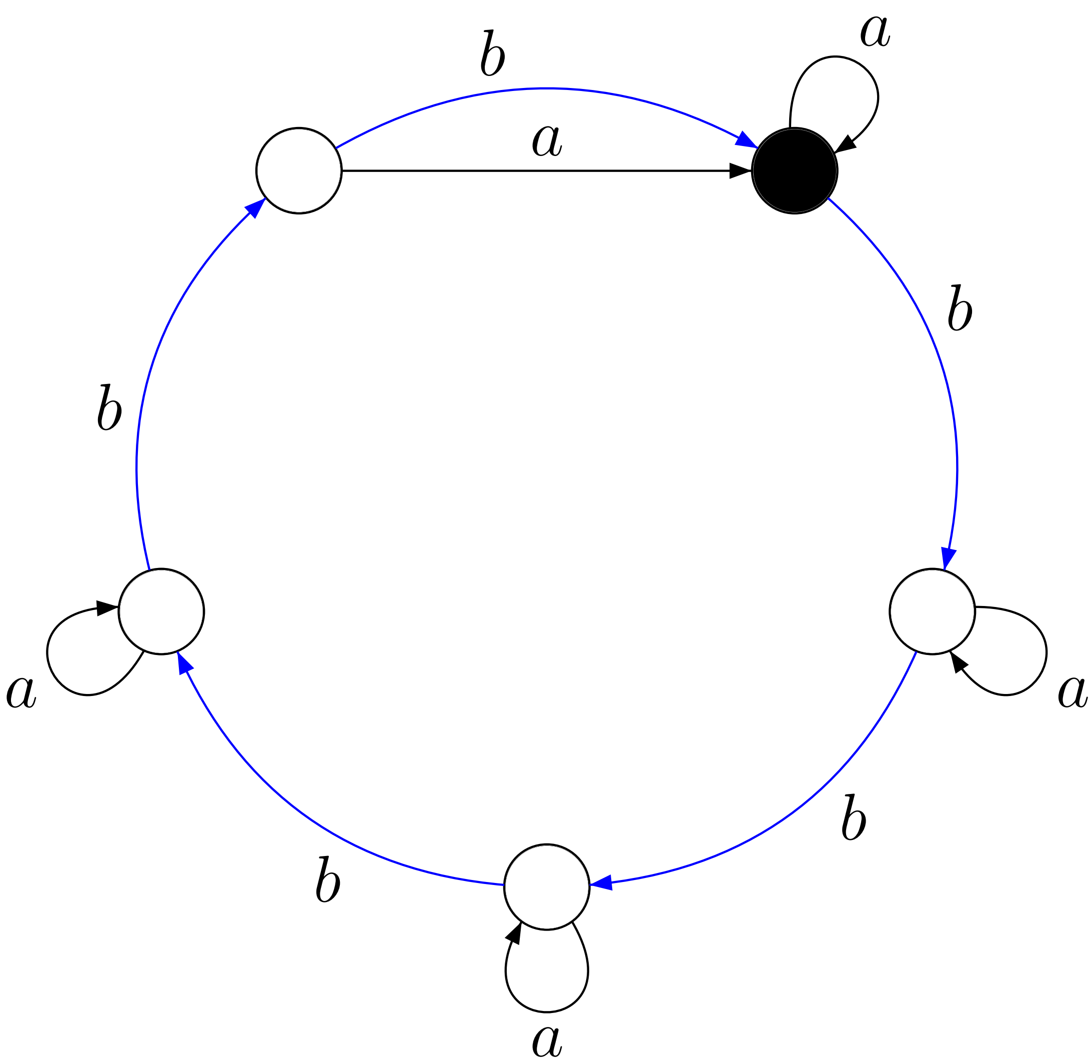
$abbbbabbbbabbbba$

reset word



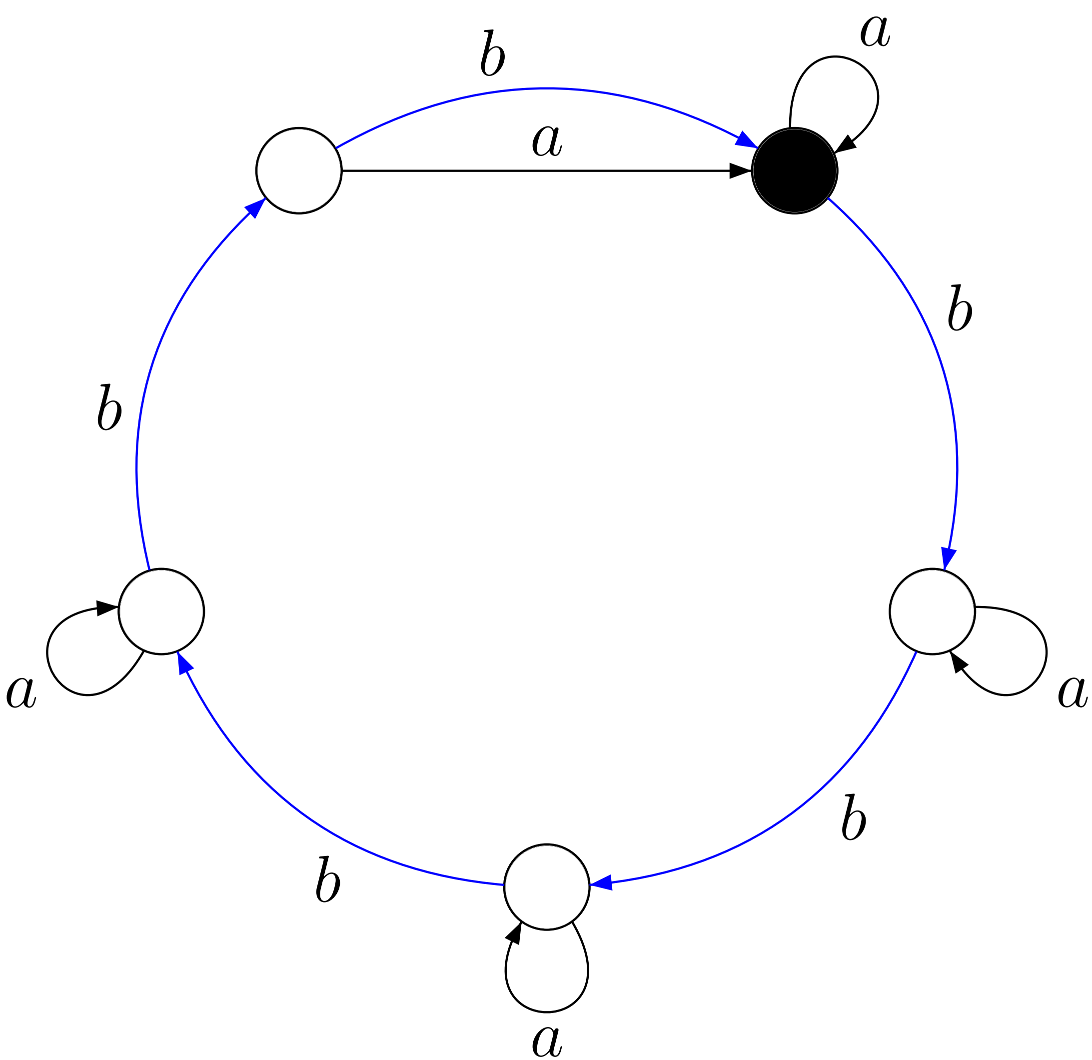
$abbbbabbbbabbbba$

reset word



$abbbbabbbbabbbb\textcolor{red}{a}$

reset word



$abbbbabbbbabbbb\color{red}{a}$

reset threshold 16

The problem of reset threshold

Problem `RT`: given an automaton $\mathscr{A}$, find its reset threshold.

The problem of reset threshold

Problem RT: given an automaton $\mathscr{A}$, find its reset threshold.

The problem is $FP^{NP[log]}$-complete [J.Olschewski, M.Ummels, 2010].

The problem of reset threshold

Problem RT: given an automaton $\mathscr{A}$, find its reset threshold.

The problem is $FP^{NP[log]}$-complete [J.Olschewski, M.Ummels, 2010].

For a fixed constant C there is no $C \cdot \log(n)$-approximation algorithm for RT (unless $P = NP$) [M.Gerbush, B,Heeringa, 2011],

The problem of reset threshold

Problem RT: given an automaton $\mathscr{A}$, find its reset threshold.

The problem is $FP^{NP[log]}$-complete [J.Olschewski, M.Ummels, 2010].

For a fixed constant C there is no $C \cdot \log(n)$-approximation algorithm for $\mathtt{RT}$ (unless $P = NP$) [M.Gerbush, B,Heeringa, 2011],

even if we restrict ourselves to two-letter automata [M.Berlinkov, 2013].

Class of algorithms

$SYNCH-SUBSET(k)$

Input: Synchronizing automaton $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$

$S = Q$

$u = \varepsilon$

Until $|S| == 1$ do

    If $|S| \geq k$:

        Choose a set $P \subseteq S$ s.t. $|P| = k$

    Else:

        $P = S$

    Choose a word v s.t. $|P.v| = 1$

    $u = u \cdot v$

    $S = S.v$

Return length of u

Greedy choices

We say that the choice of word v is *greedy* if for any other word v' with the property $|P.v'| = 1$ we have $|v| \leq |v'|$.

Greedy choices

We say that the choice of word v is *greedy* if for any other word v' with the property $|P.v'| = 1$ we have $|v| \leq |v'|$.

We say that the choice of set P is *greedy* if for any other subset P' and a word v' such that $|P'.v'| = 1$ there is a word v such that $|P.v| = 1$ and $|v| \leq |v'|$.

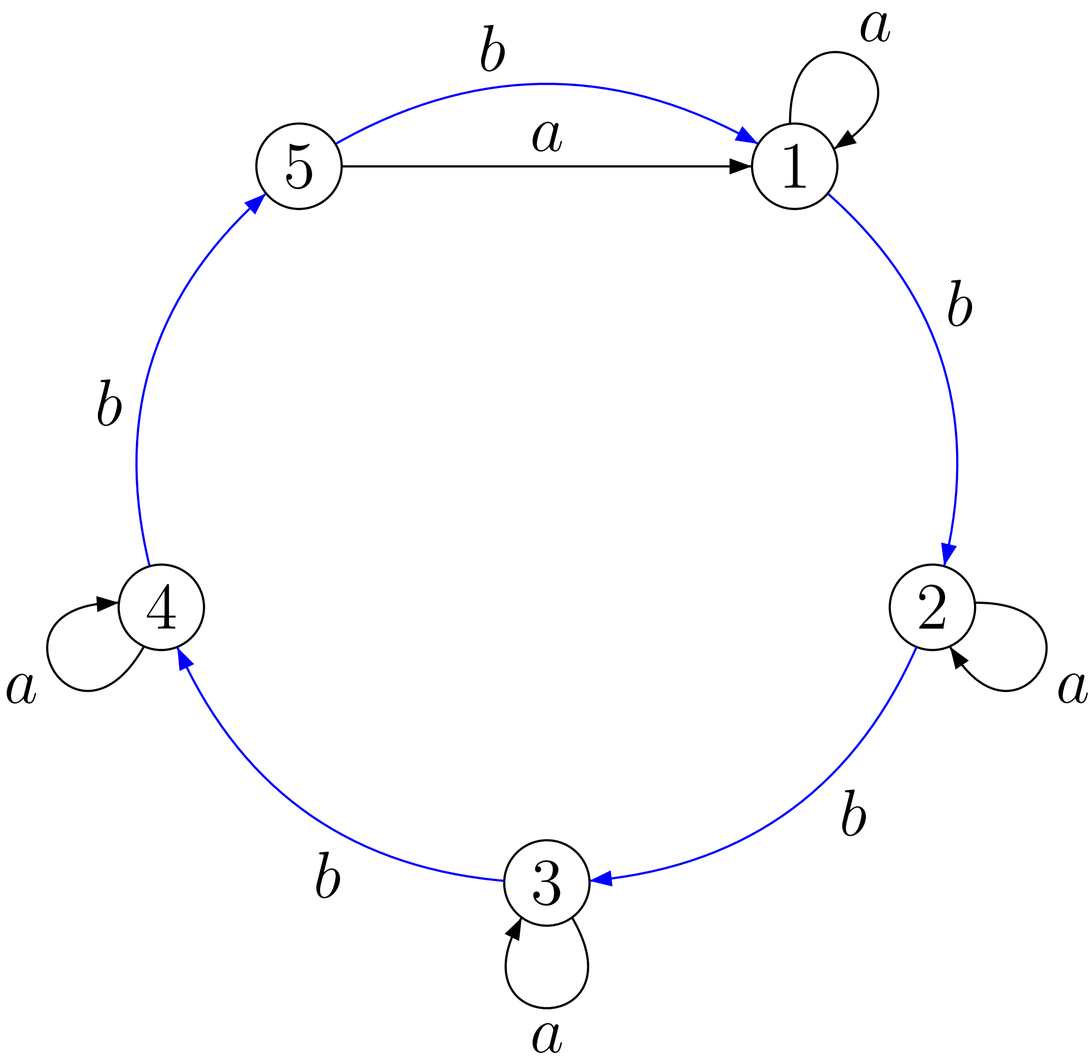
Greedy choices

We say that the choice of word v is *greedy* if for any other word v' with the property $|P.v'| = 1$ we have $|v| \leq |v'|$.

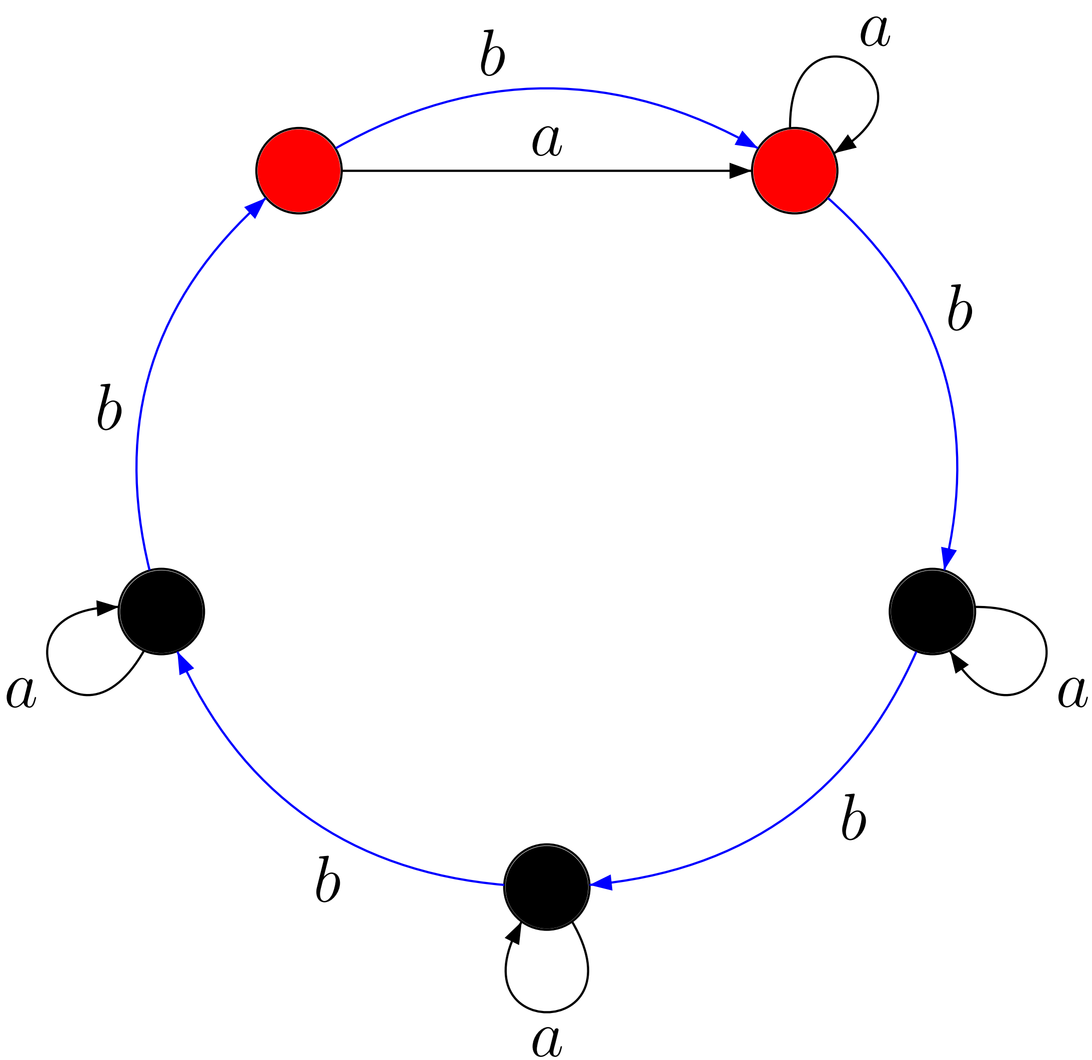
We say that the choice of set P is *greedy* if for any other subset P' and a word v' such that $|P'.v'| = 1$ there is a word v such that $|P.v| = 1$ and $|v| \leq |v'|$.

Greedy choices can be done after polynomial-time computation.

“greedy” reset word (k=2)

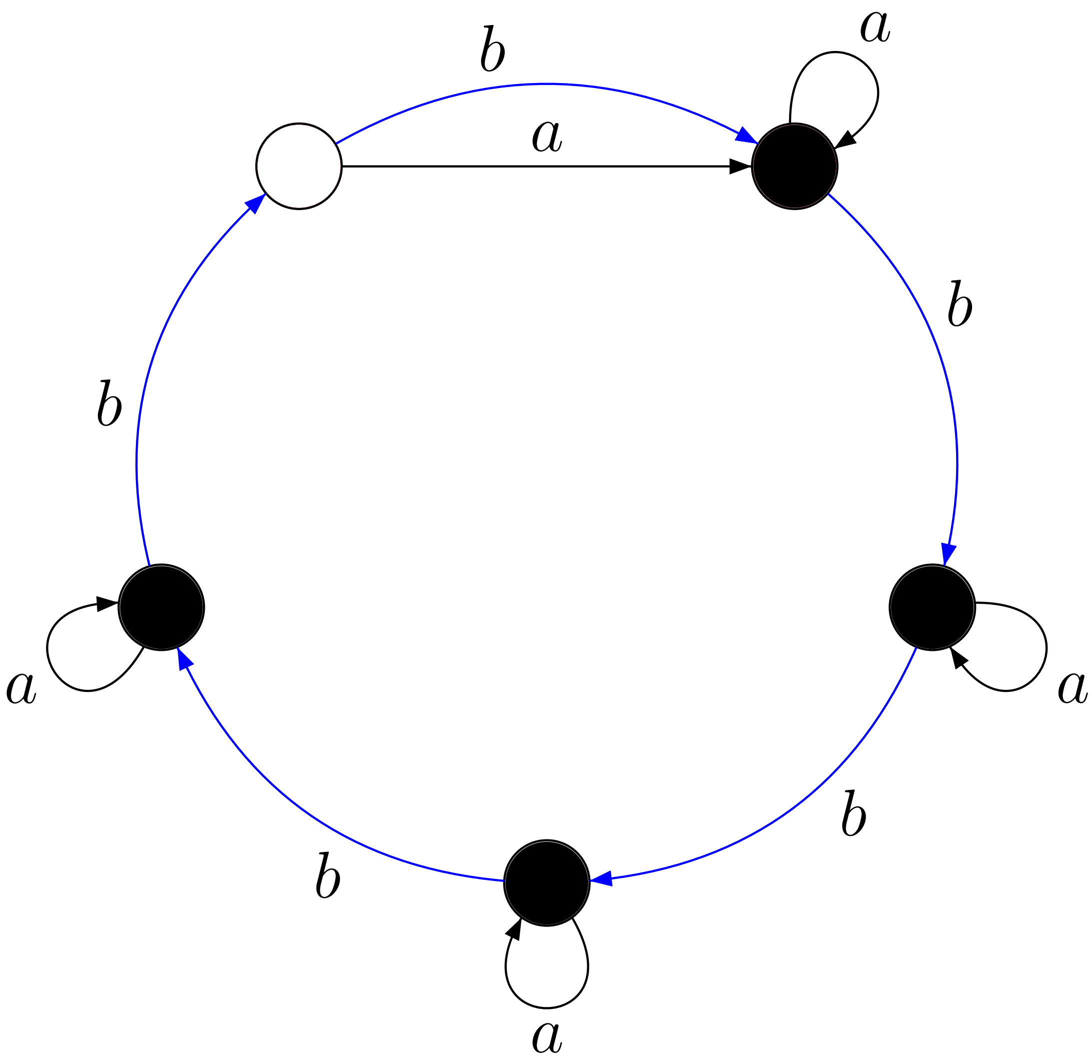


“greedy” reset word (k=2)



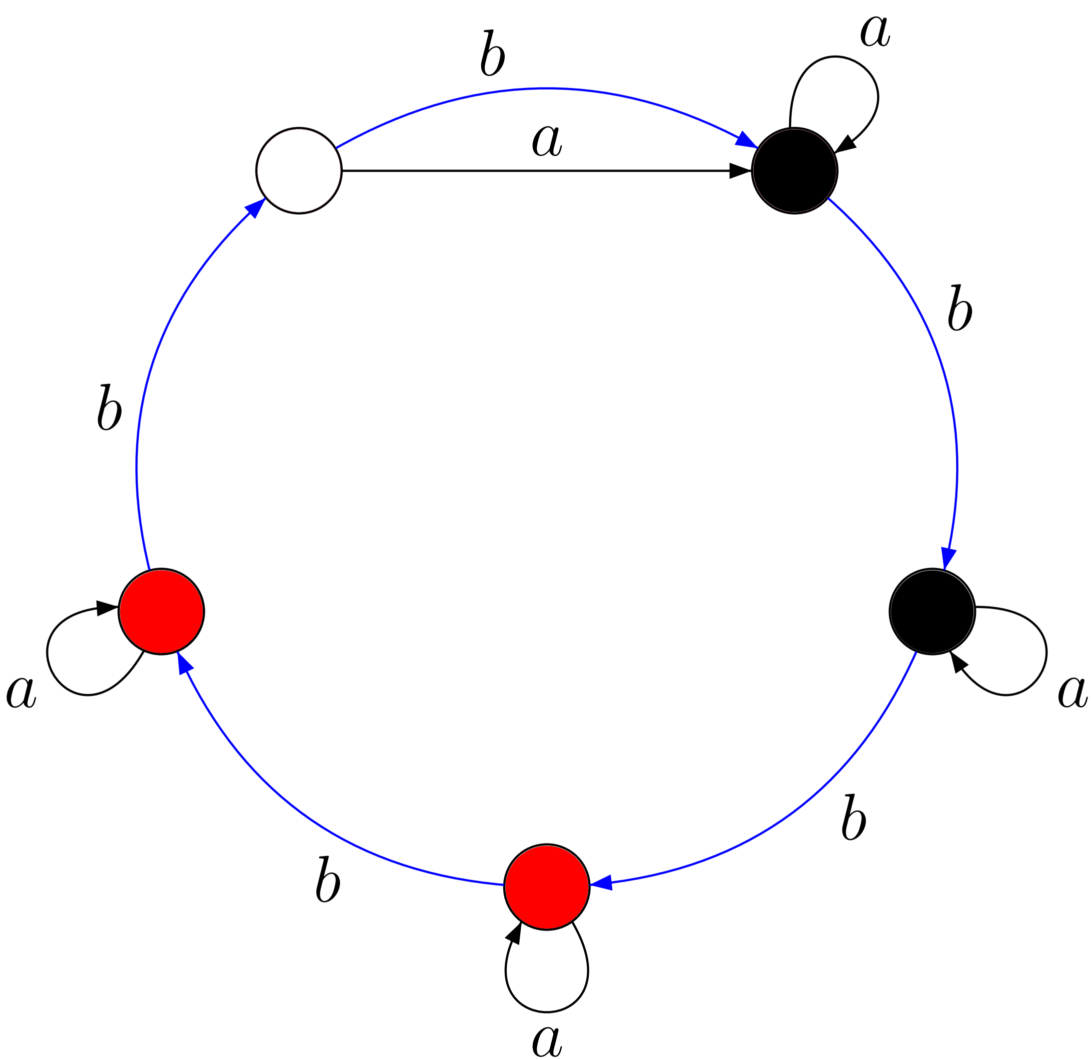
a

“greedy” reset word (k=2)



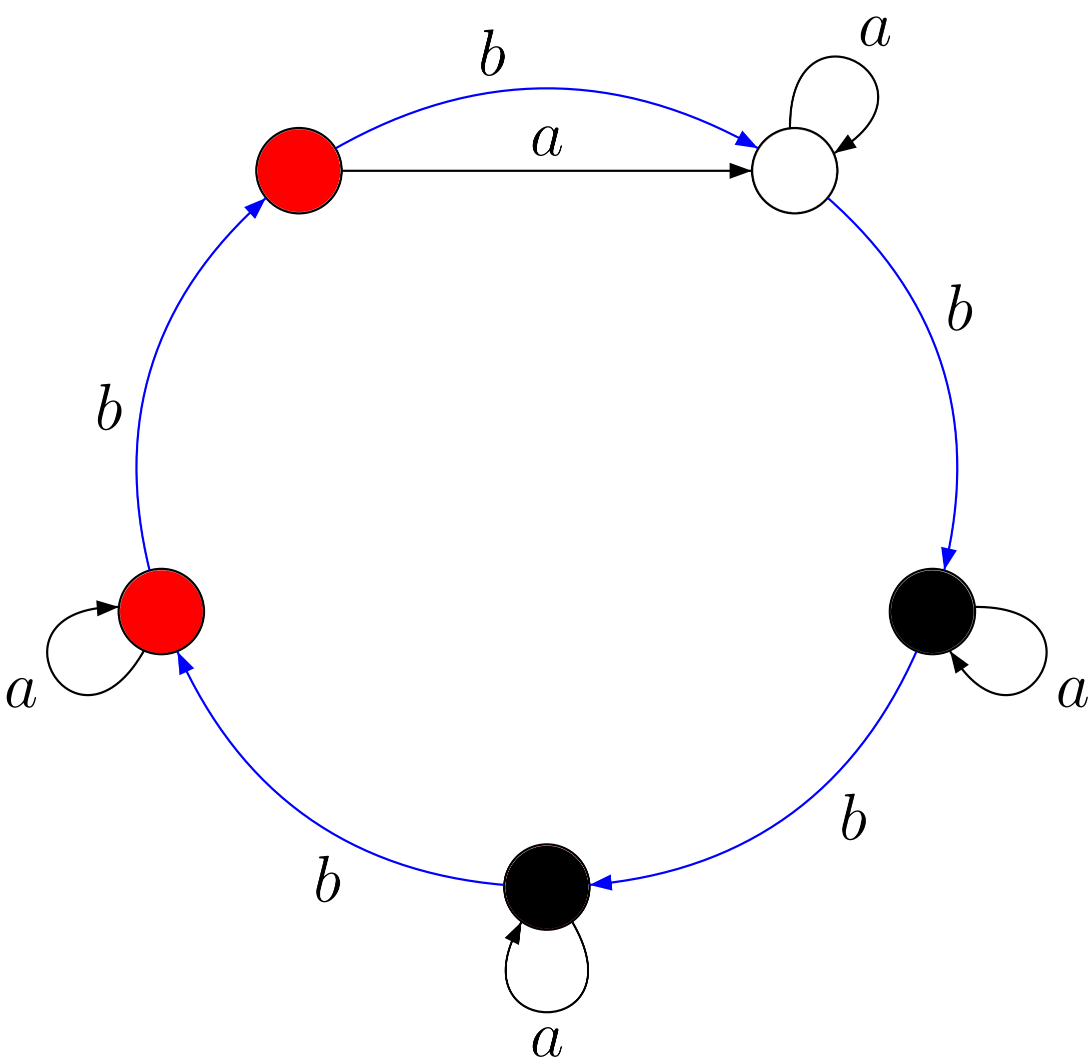
a

“greedy” reset word (k=2)



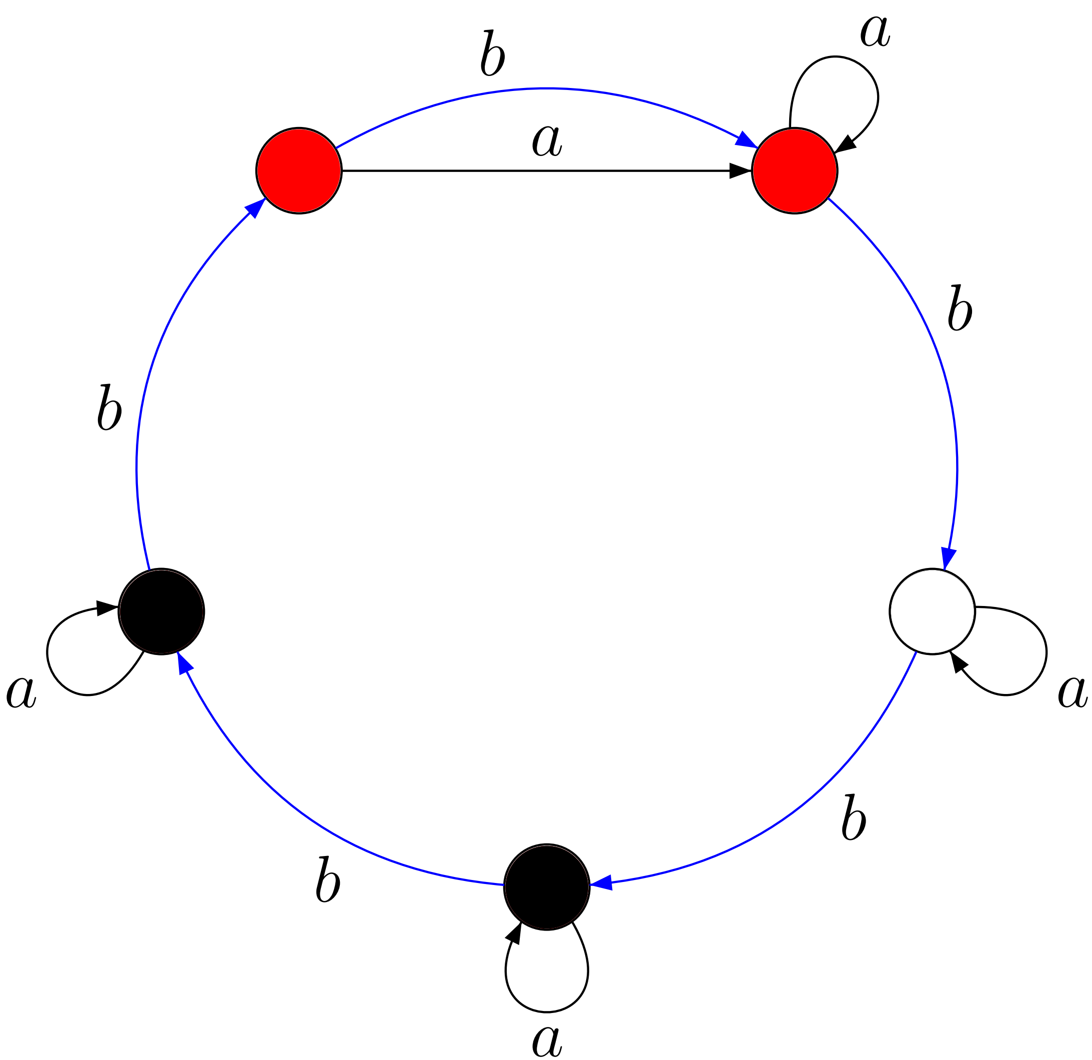
$\color{red}{a}bba$

“greedy” reset word (k=2)



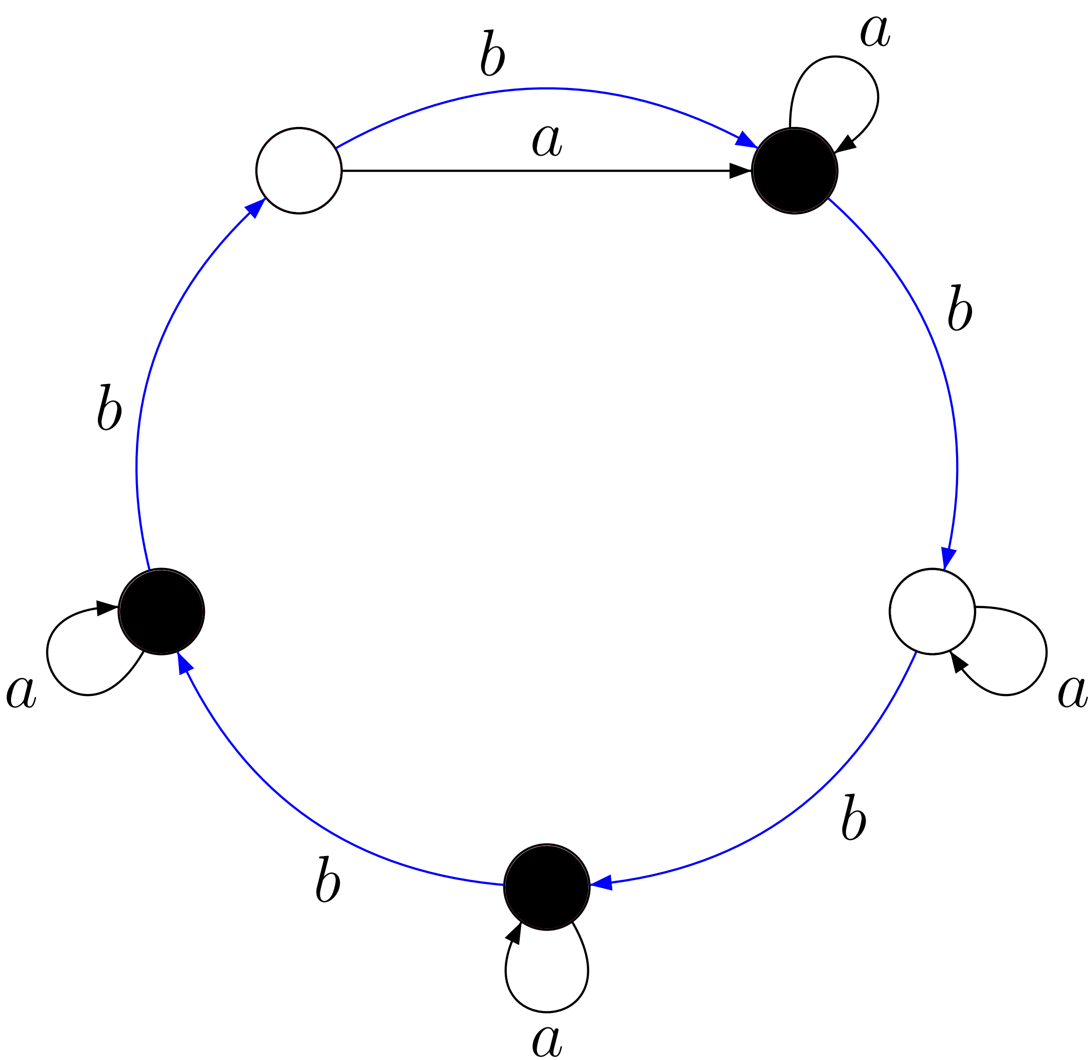
$abba$

“greedy” reset word (k=2)



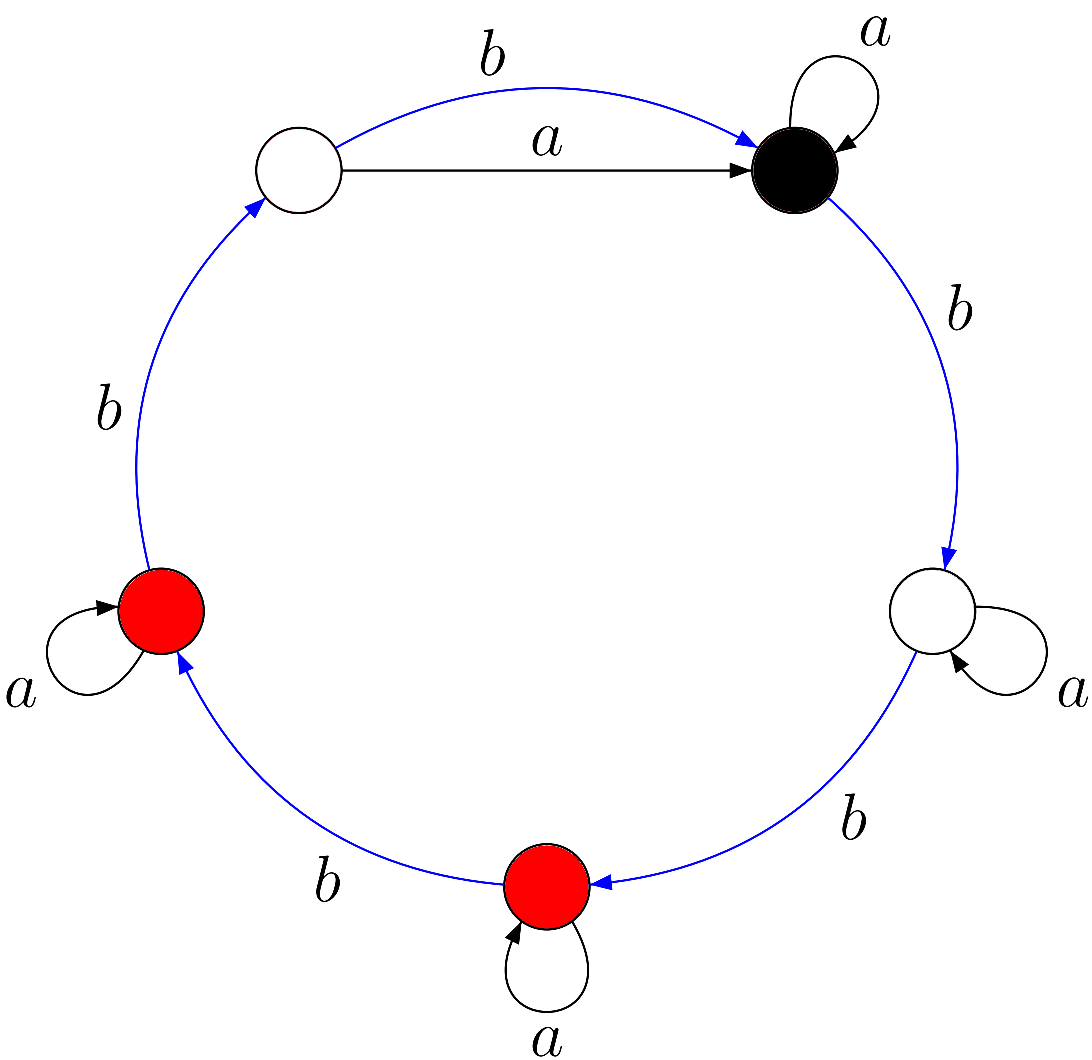
$abba$

“greedy” reset word (k=2)



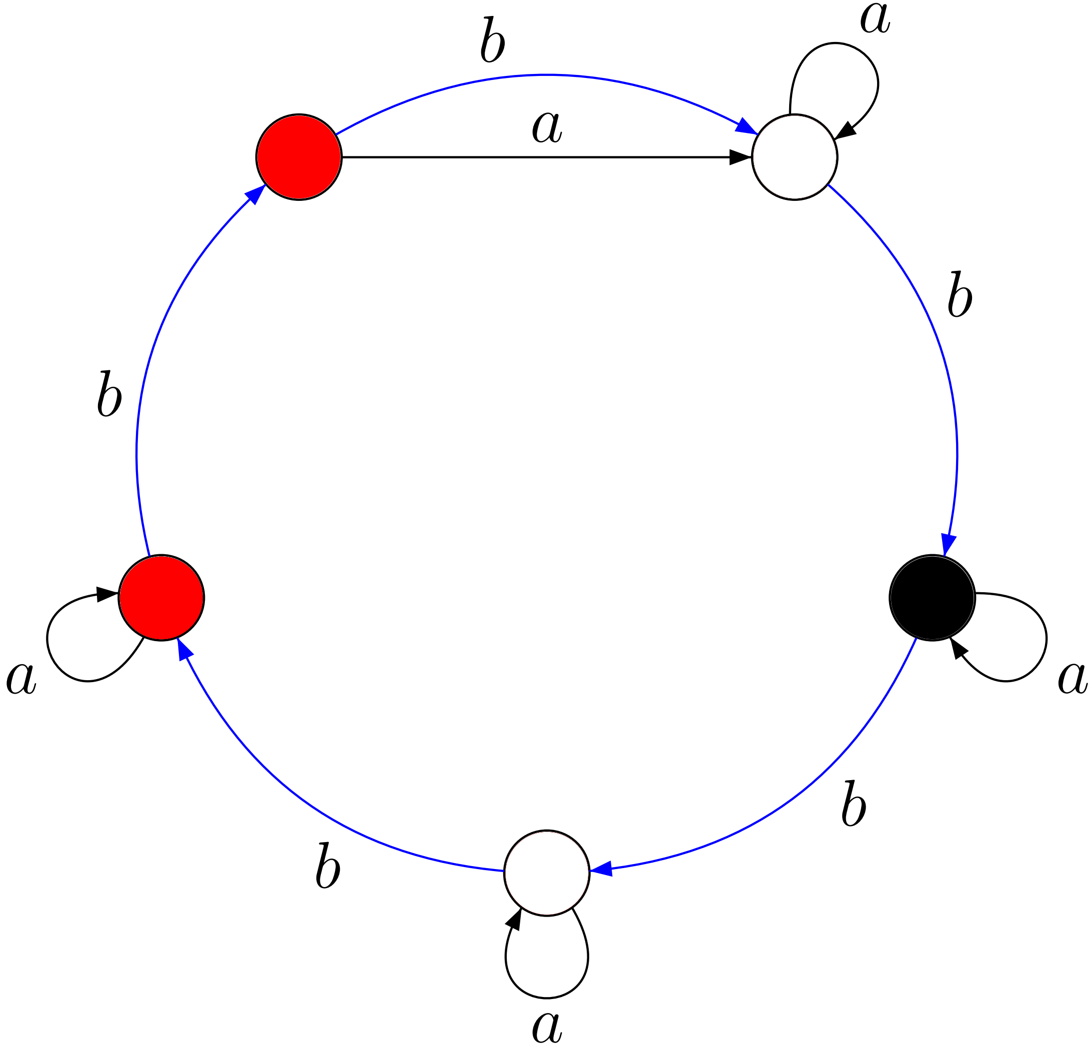
$ab b\textcolor{red}{a}$

“greedy” reset word (k=2)



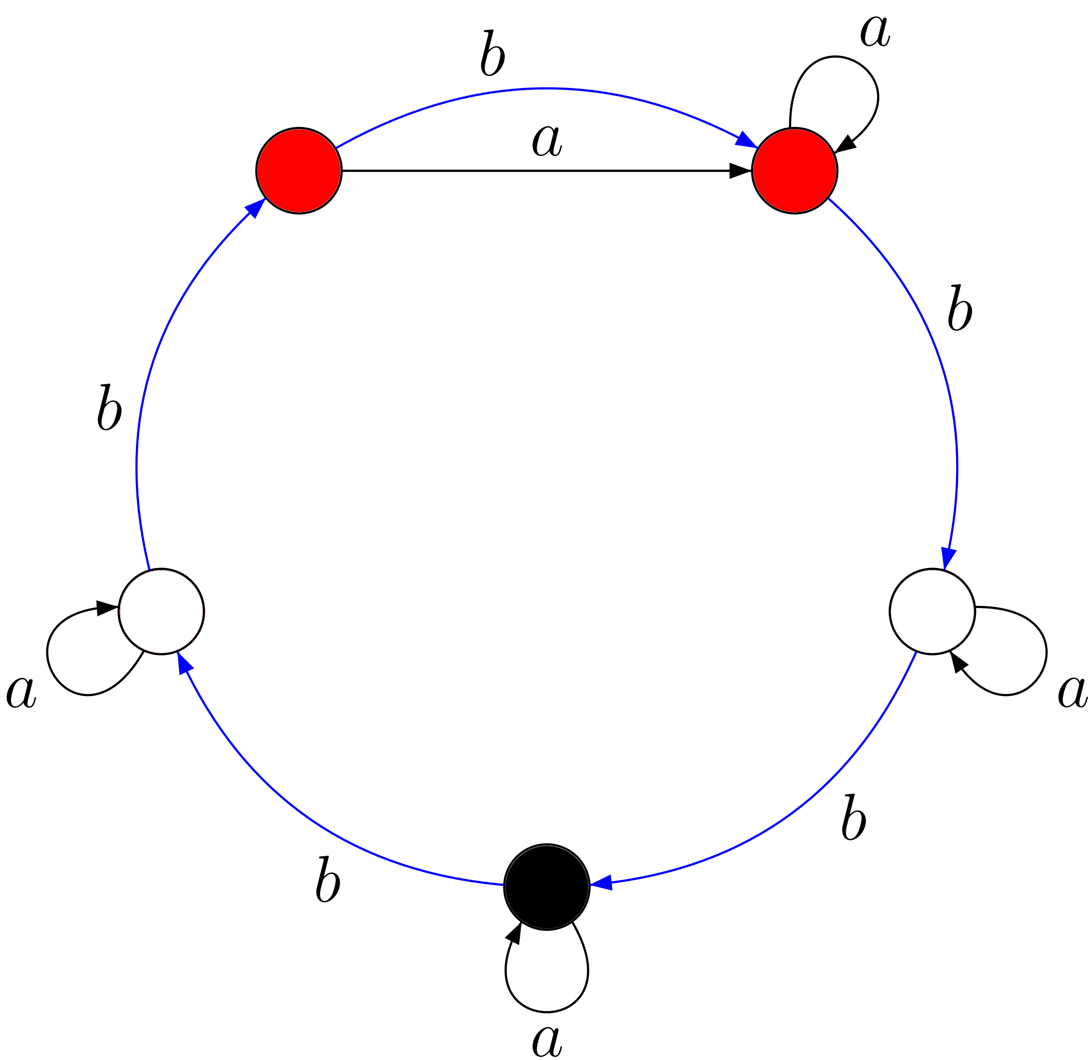
$abb{\color{red}a}bba$

“greedy” reset word (k=2)



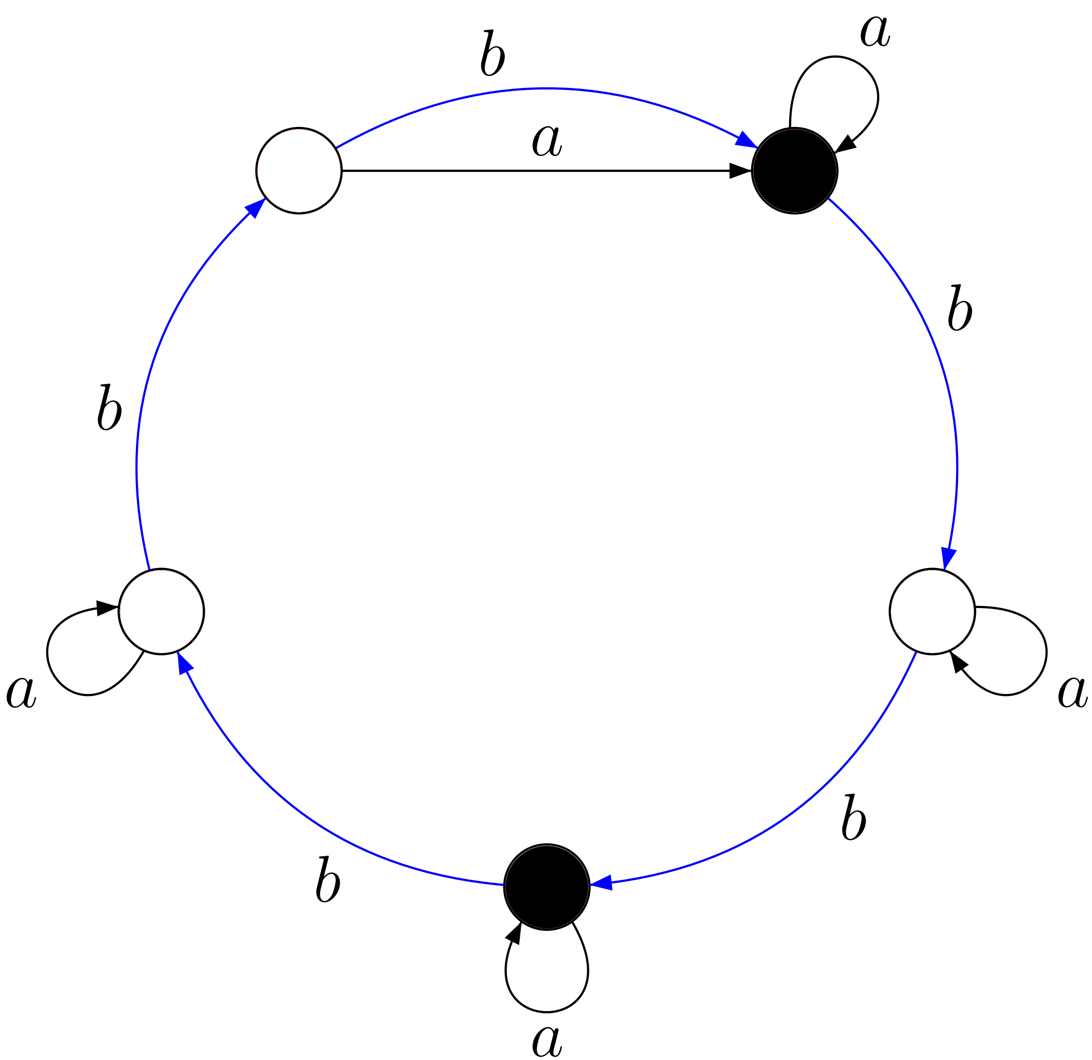
$abbababba$

“greedy” reset word (k=2)



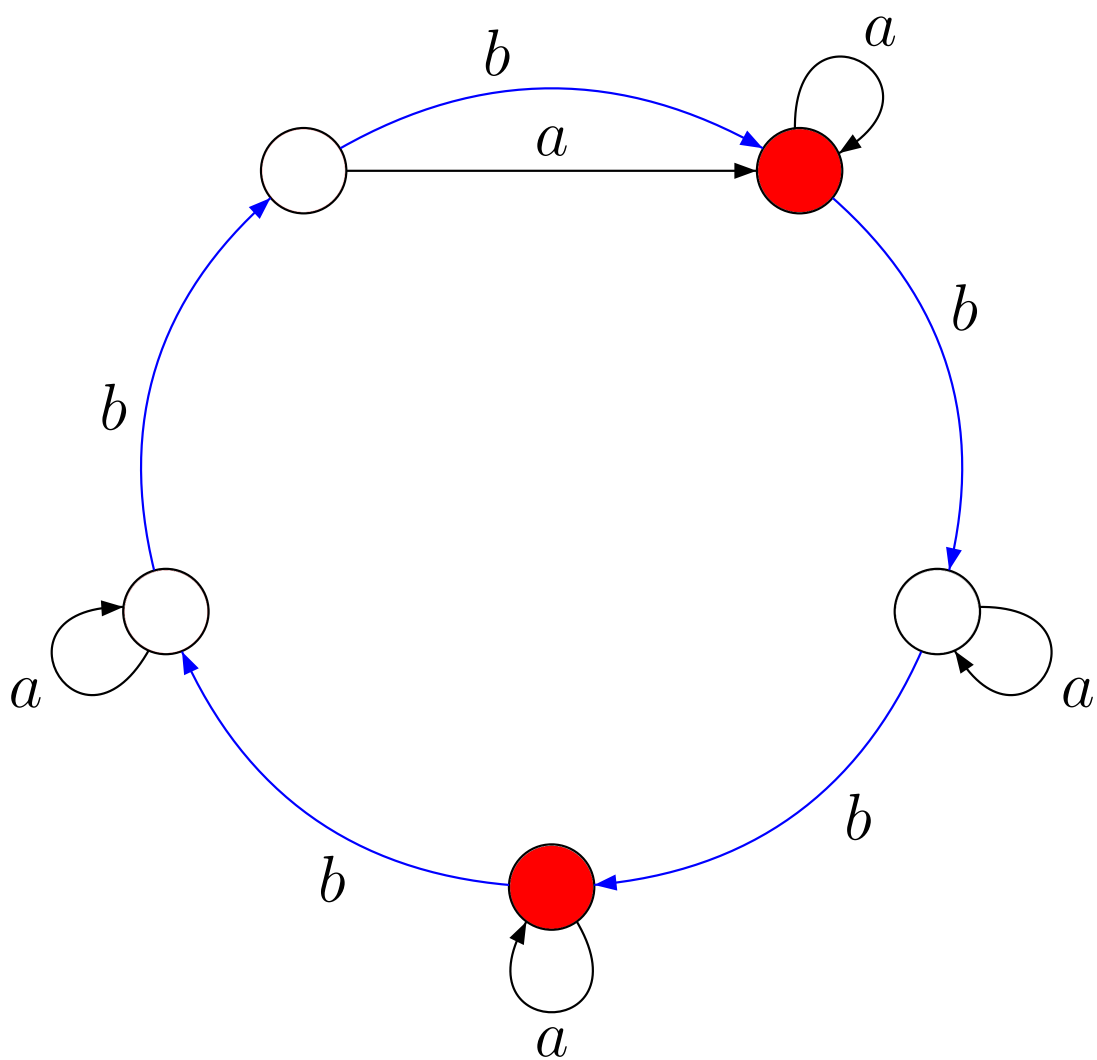
$abbab\textcolor{red}{b}a$

“greedy” reset word (k=2)



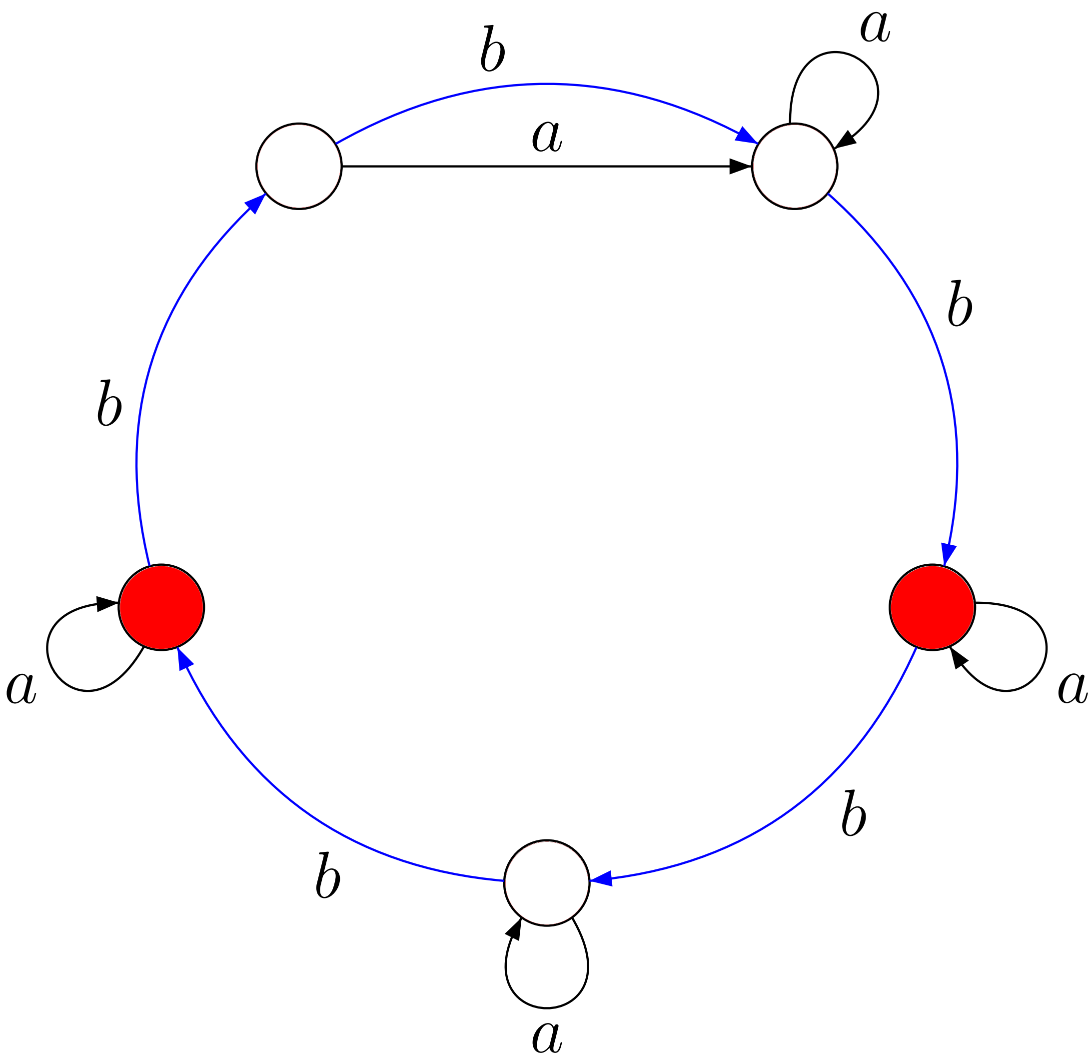
$ab b a b b {\color{red}a}$

“greedy” reset word (k=2)



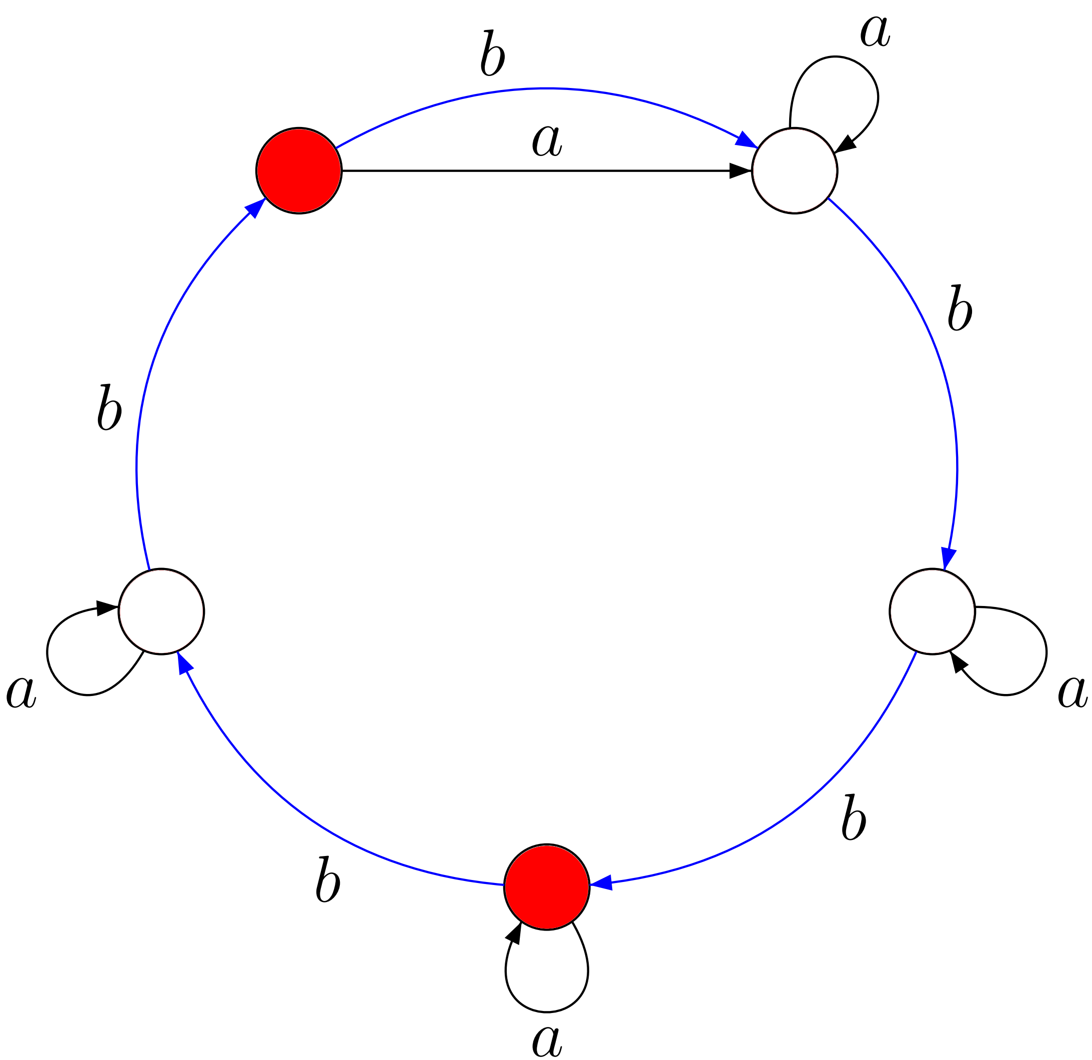
$abbabbabbbbabbbba$

“greedy” reset word (k=2)



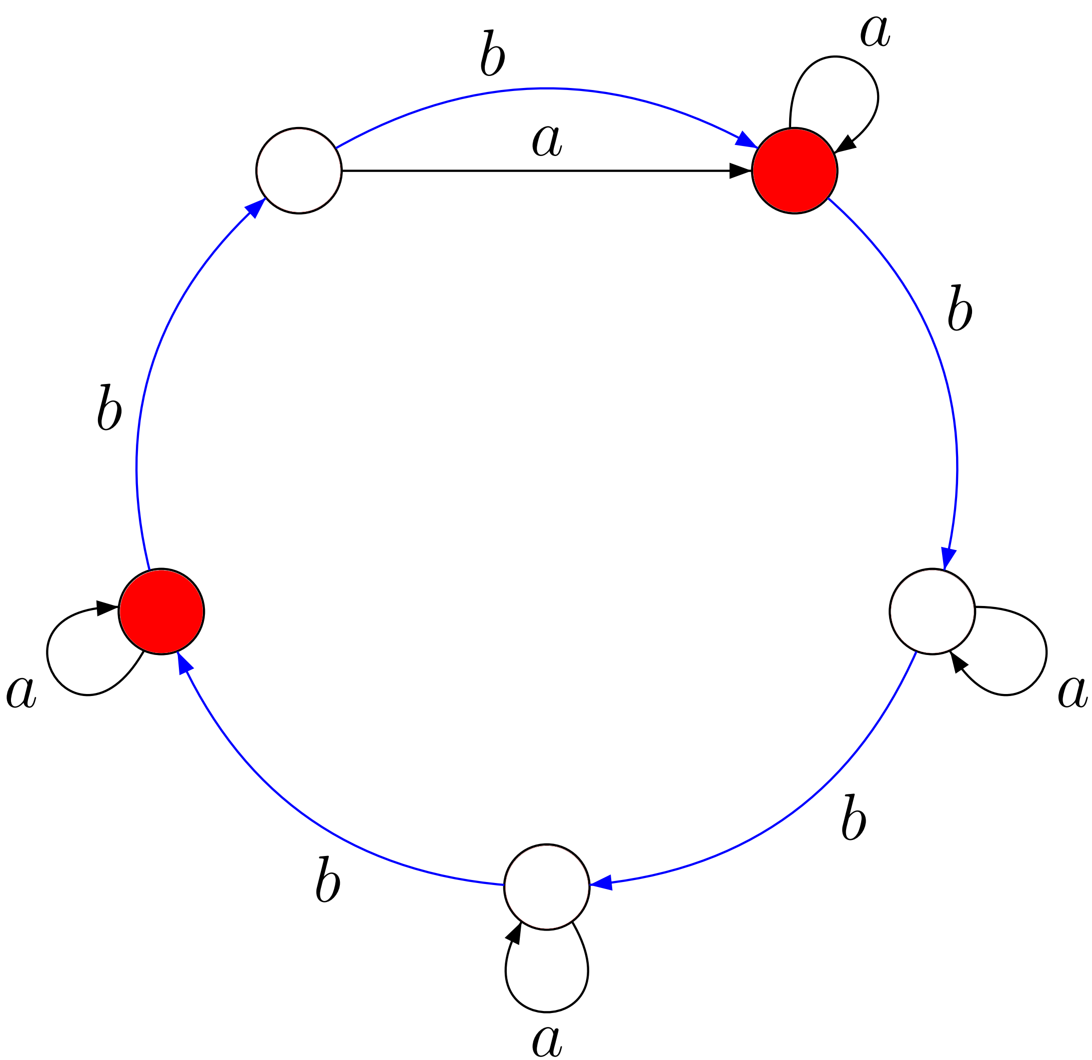
$abbabbabbbbabbbba$

“greedy” reset word (k=2)



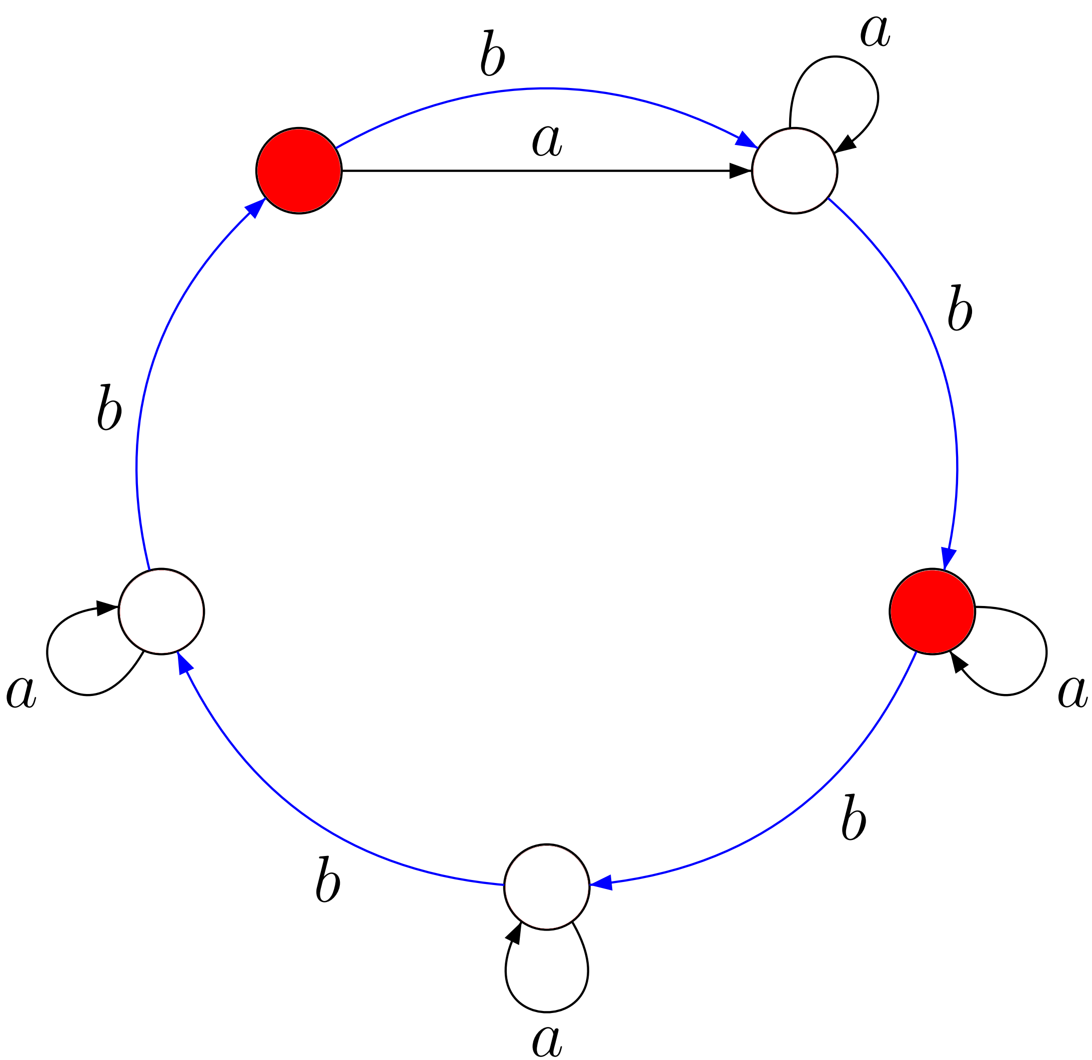
$abbabbab\textcolor{red}{b}bbabbbba$

“greedy” reset word (k=2)



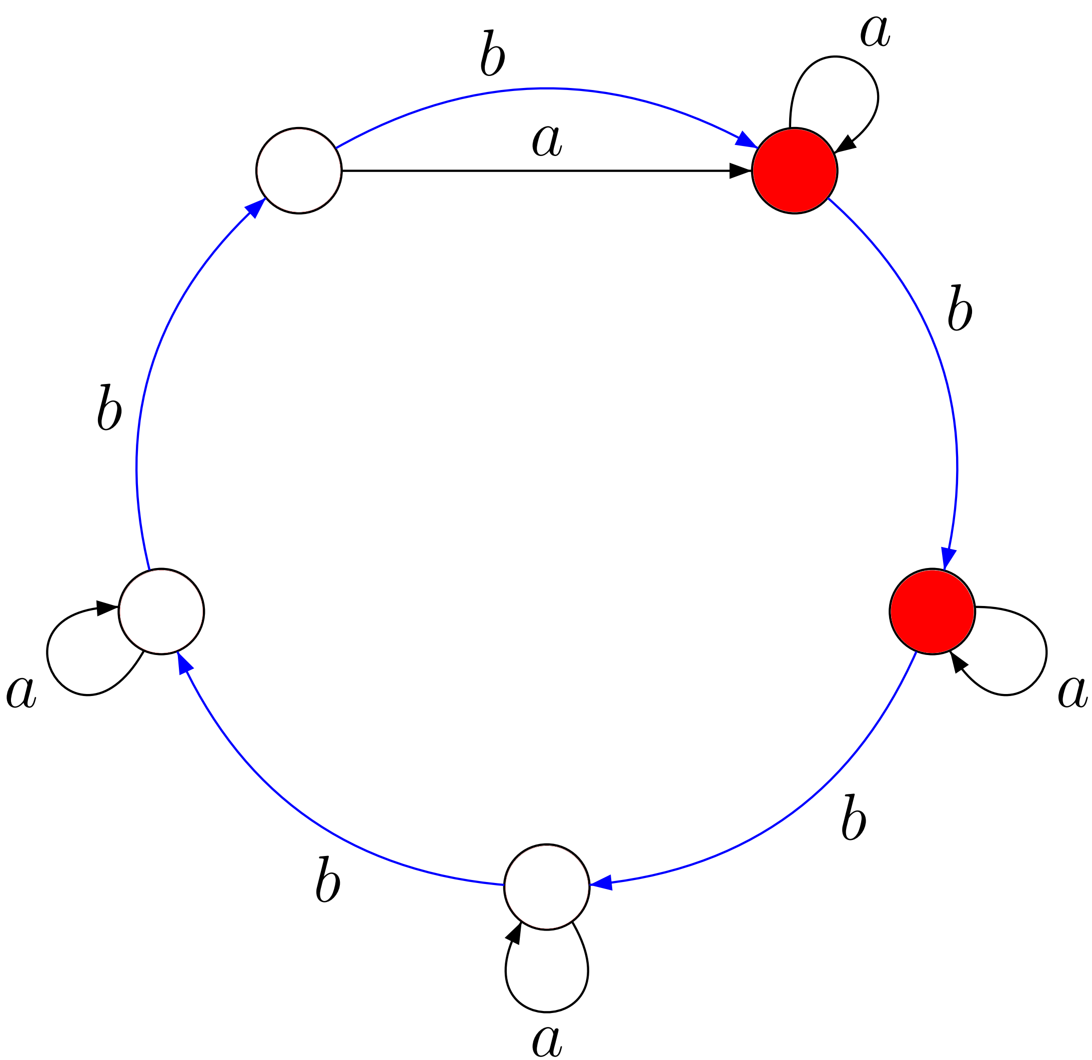
$abbabbabb\color{red}{b}\color{black}{babbbba}$

“greedy” reset word (k=2)



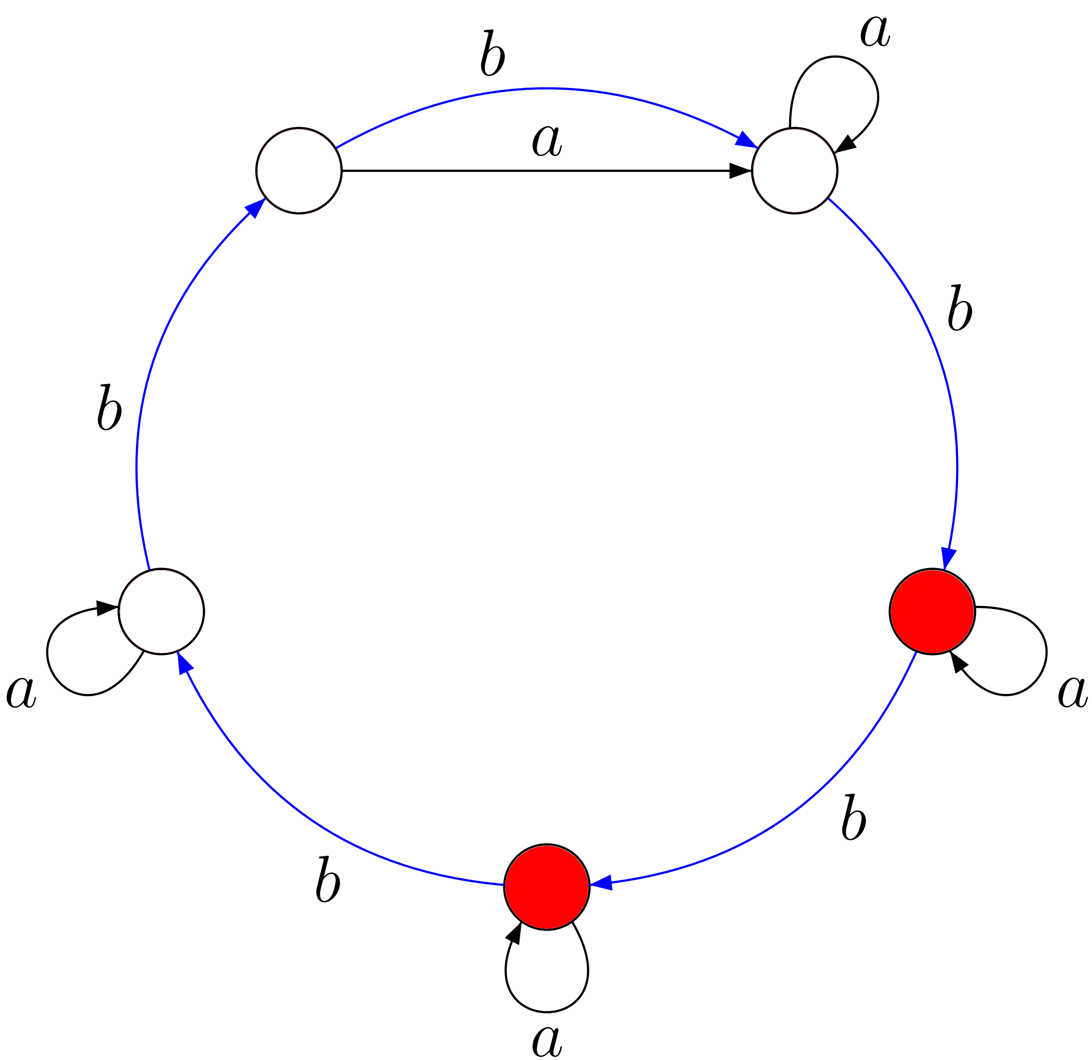
$abbabbabbb\textcolor{red}{b}abbbba$

“greedy” reset word (k=2)



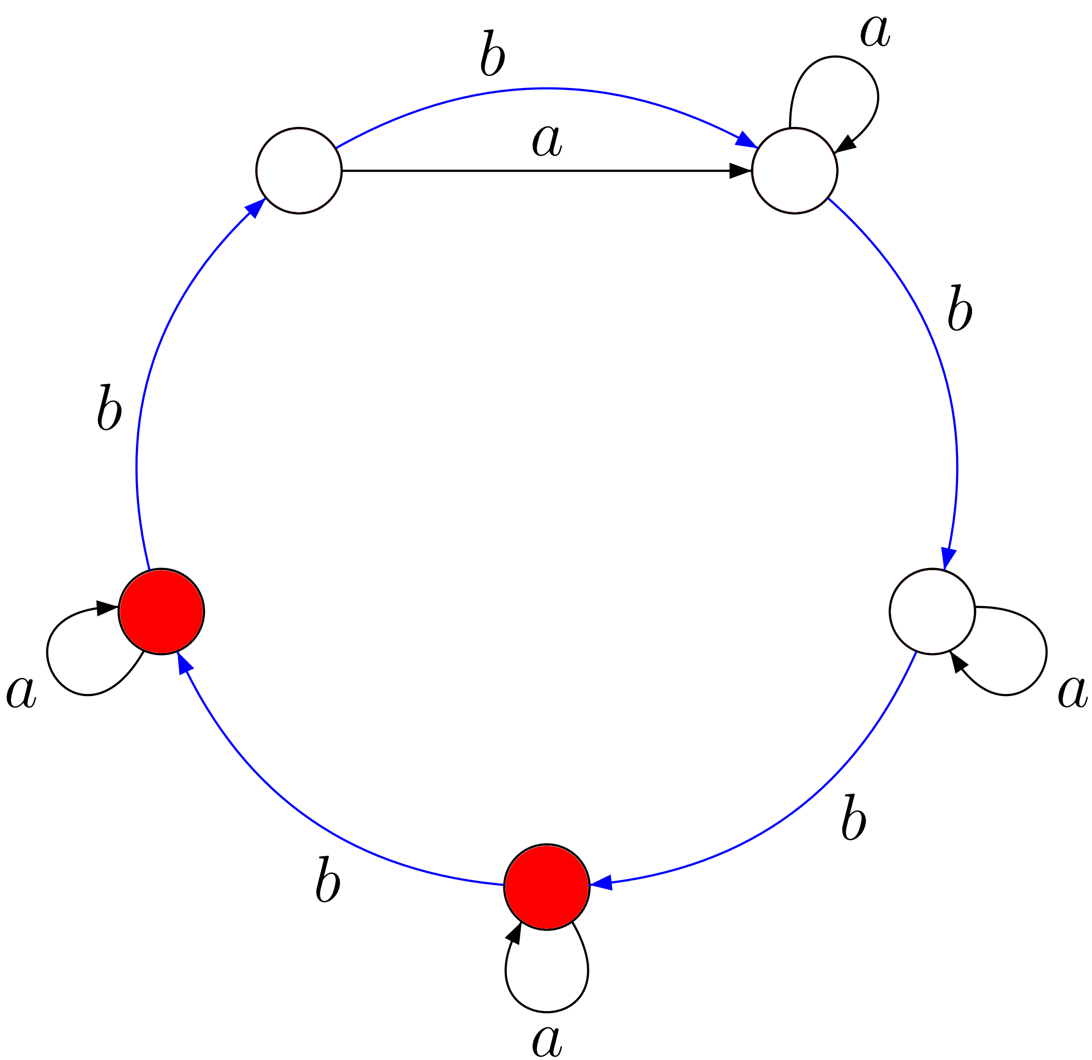
$abbabbabbbb\textcolor{red}{a}bbbba$

“greedy” reset word (k=2)



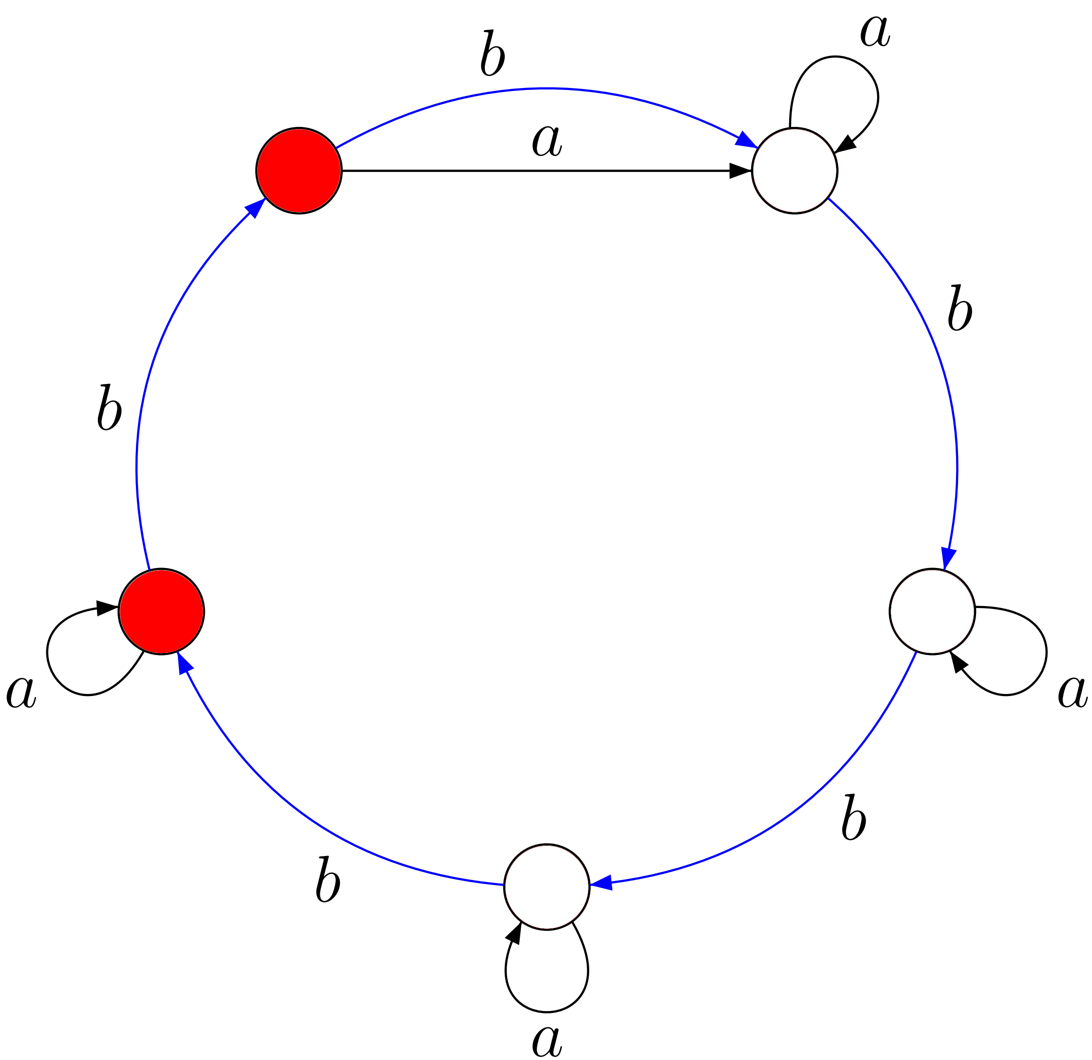
$abbabbabbbbabbbba$

“greedy” reset word (k=2)



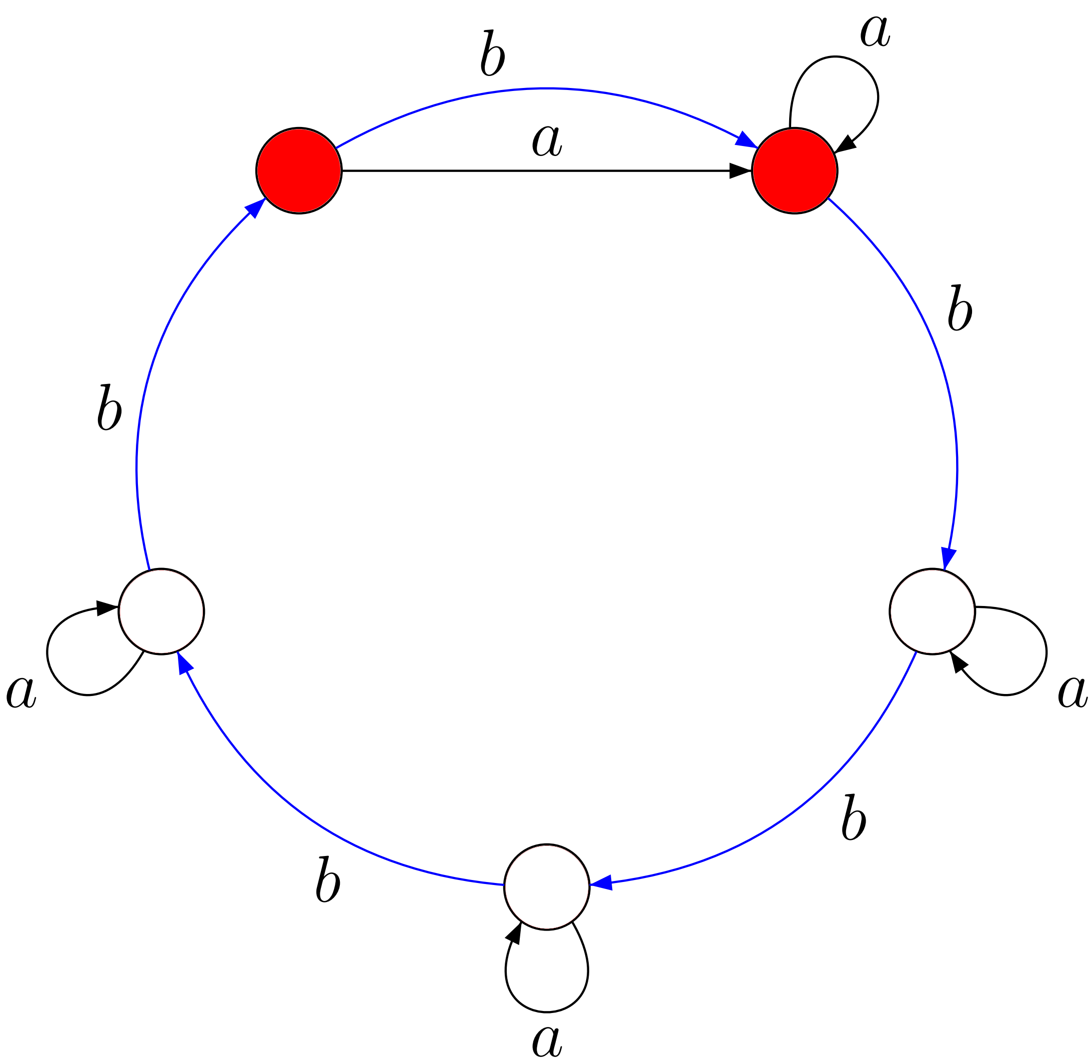
$abbabbabbbbab\textcolor{red}{b}bba$

“greedy” reset word (k=2)



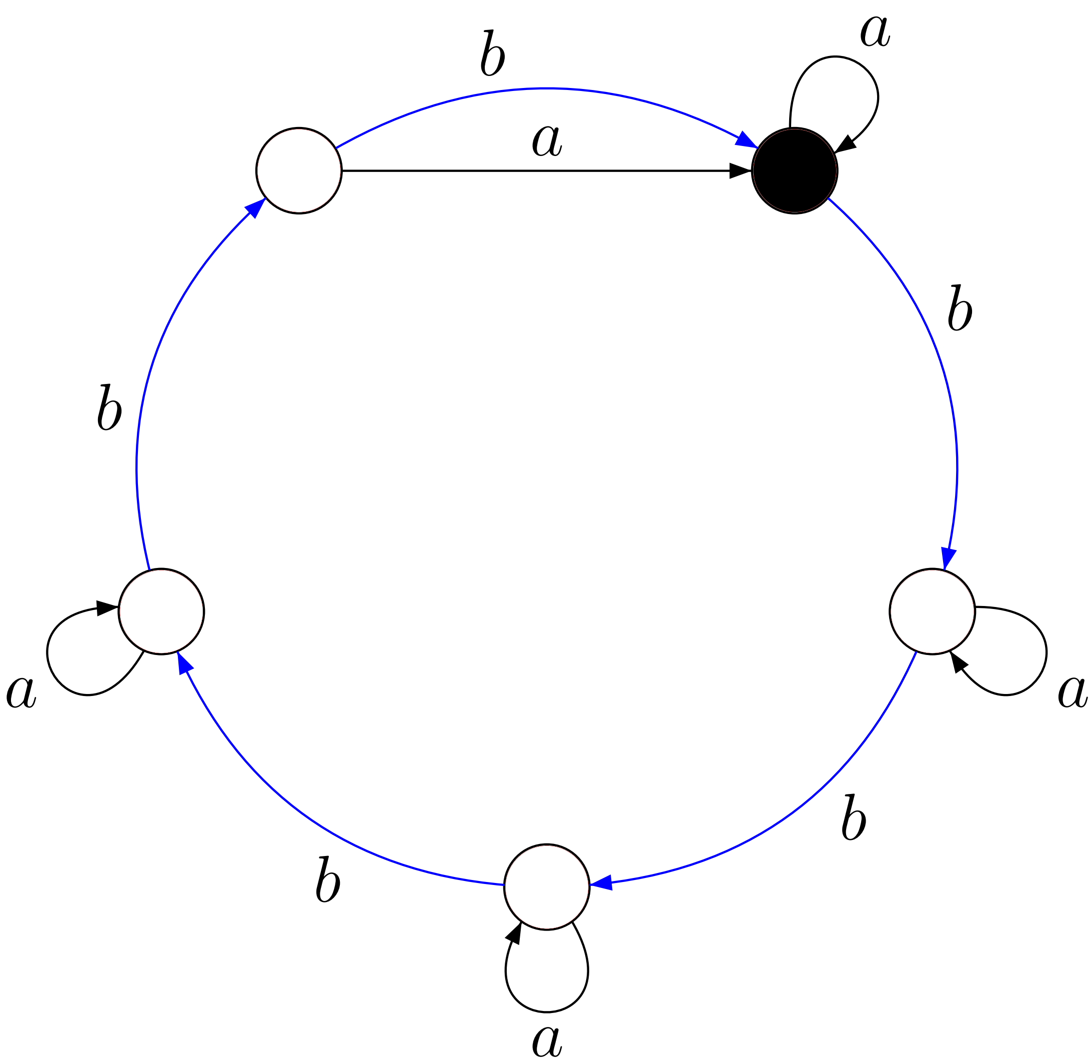
$abbabbabbbbabb\textcolor{red}{b}ba$

“greedy” reset word (k=2)



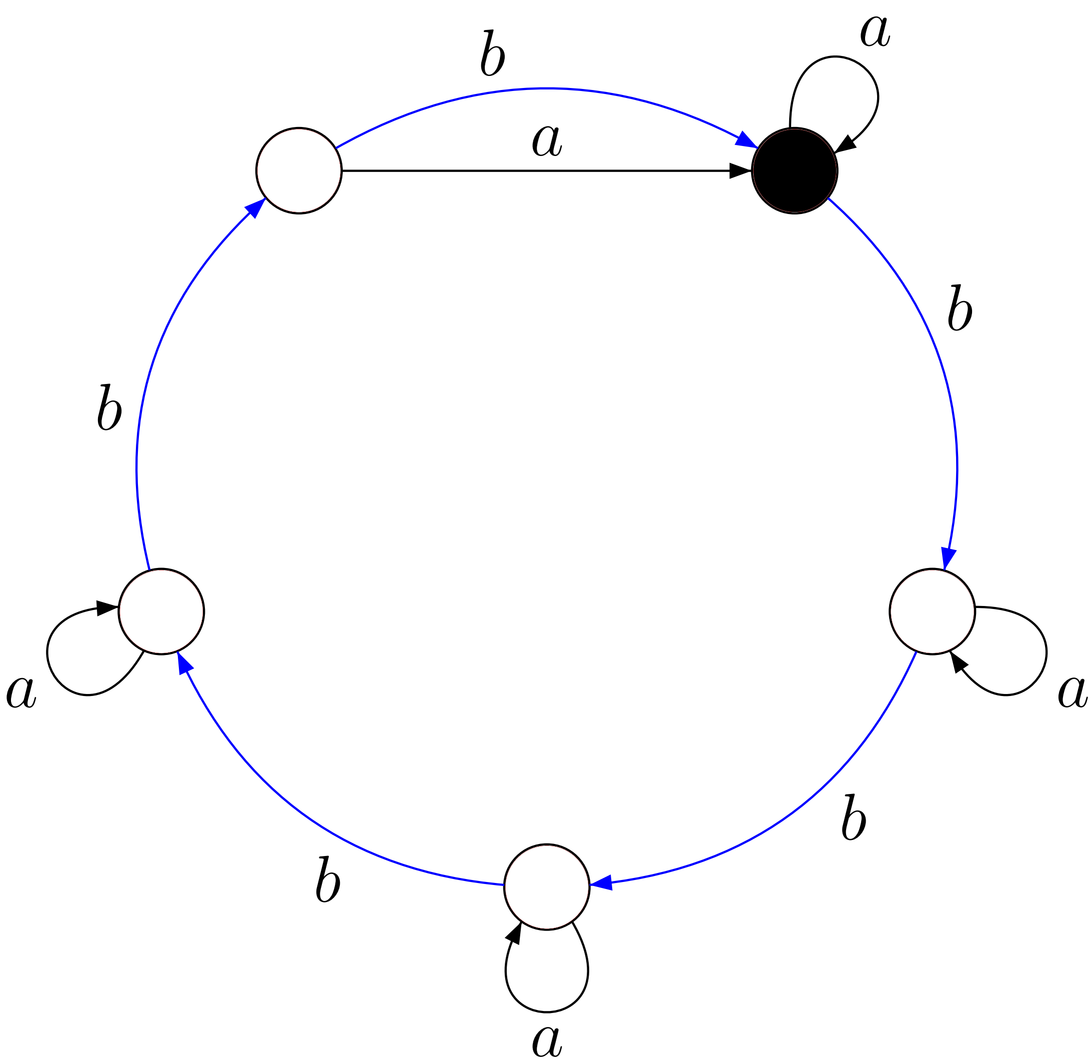
$abbabbabbbbabbbba$

"greedy" reset word (k=2)



$abbabbabbbbabbbb\textcolor{red}{a}$

“greedy” reset word (k=2)



$abbabbabbbbabbbb\color{red}{a}$

approximation ratio $\frac{17}{16}$

Approximation ratio

Approximation ratio of $SYNCH - SUBSET(k)$ with a greedy choice of word is at most $\lceil \frac{n-1}{k-1} \rceil$. [M.Gerbush, B,Heeringa, 2011]

Approximation ratio

Approximation ratio of $SYNCH - SUBSET(k)$ with a greedy choice of word is at most $\lceil \frac{n-1}{k-1} \rceil$. [M.Gerbush, B,Heeringa, 2011]

Proof. Let w be the shortest reset word for $\mathscr{A}$.

Approximation ratio

Approximation ratio of $SYNCH - SUBSET(k)$ with a greedy choice of word is at most $\lceil \frac{n-1}{k-1} \rceil$. [M.Gerbush, B,Heeringa, 2011]

Proof. Let w be the shortest reset word for $\mathscr{A}$.

No matter the set P we have $|P.w| = 1$. Hence $|v| \leq |w|$.

Approximation ratio

Approximation ratio of $SYNCH - SUBSET(k)$ with a greedy choice of word is at most $\lceil \frac{n-1}{k-1} \rceil$. [M.Gerbush, B,Heeringa, 2011]

Proof. Let w be the shortest reset word for $\mathscr{A}$.

No matter the set P we have $|P.w| = 1$. Hence $|v| \leq |w|$.

We apply it at most $\lceil \frac{n-1}{k-1} \rceil$ times. Therefore $|u| \leq |w| \cdot \lceil \frac{n-1}{k-1} \rceil$.

Approximation ratio

The bound $\lceil\frac{n-1}{k-1}\rceil$ of approximation ratio of $SYNCH-SUBSET(k)$ with a greedy choice of word is tight. (Modification of construction M.Gerbush, B,Heeringa, 2011)

Approximation ratio

The bound $\lceil \frac{n-1}{k-1} \rceil$ of approximation ratio of $SYNCH - SUBSET(k)$ with a greedy choice of word is tight. (Modification of construction M.Gerbush, B,Heeringa, 2011)

Proof For a given n consider $\mathscr{A}$ with $Q = \{0, ..., n-1\}$ $\Sigma = \{a_X | X \subseteq Q,\ |X| = k\} \cup \{a_Q\}$.

Approximation ratio

The bound $\lceil \frac{n-1}{k-1} \rceil$ of approximation ratio of $SYNCH-SUBSET(k)$ with a greedy choice of word is tight. (Modification of construction M.Gerbush, B,Heeringa, 2011)

Proof For a given n consider $\mathscr{A}$ with $Q = \{0, ..., n-1\}$ $\Sigma = \{a_X | X \subseteq Q,\ |X| = k\} \cup \{a_Q\}$.

For $a_X \in \Sigma$ and $q \in Q$ let

$$\delta(q, a_X) = \begin{cases} \min(X), & if\ q \in X \\ q, & if\ q \notin X \end{cases}$$

Approximation ratio

The bound $\lceil \frac{n-1}{k-1} \rceil$ of approximation ratio of $SYNCH-SUBSET(k)$ with a greedy choice of word is tight. (Modification of construction M.Gerbush, B,Heeringa, 2011)

Proof For a given n consider $\mathscr{A}$ with $Q = \{0, ..., n-1\}$ $\Sigma = \{a_X | X \subseteq Q,\ |X| = k\} \cup \{a_Q\}$.

For $a_X \in \Sigma$ and $q \in Q$ let

$$\delta(q, a_X) = \begin{cases} \min(X), & if\ q \in X \\ q, & if\ q \notin X \end{cases}$$

The shortest reset word is a_Q which has length 1.

Approximation ratio

Proof For a given n consider $\mathscr{A}$ with $Q = \{0, ..., n-1\}$
$\Sigma = \{a_X | X \subseteq Q,\ |X| = k\} \cup \{a_Q\}$.
For $a_X \in \Sigma$ and $q \in Q$ let

$$\delta(q, a_X) = \begin{cases} \min(X), & if\ q \in X \\ q, & if\ q \notin X \end{cases}$$

The shortest reset word is a_Q which has length 1.

Approximation ratio

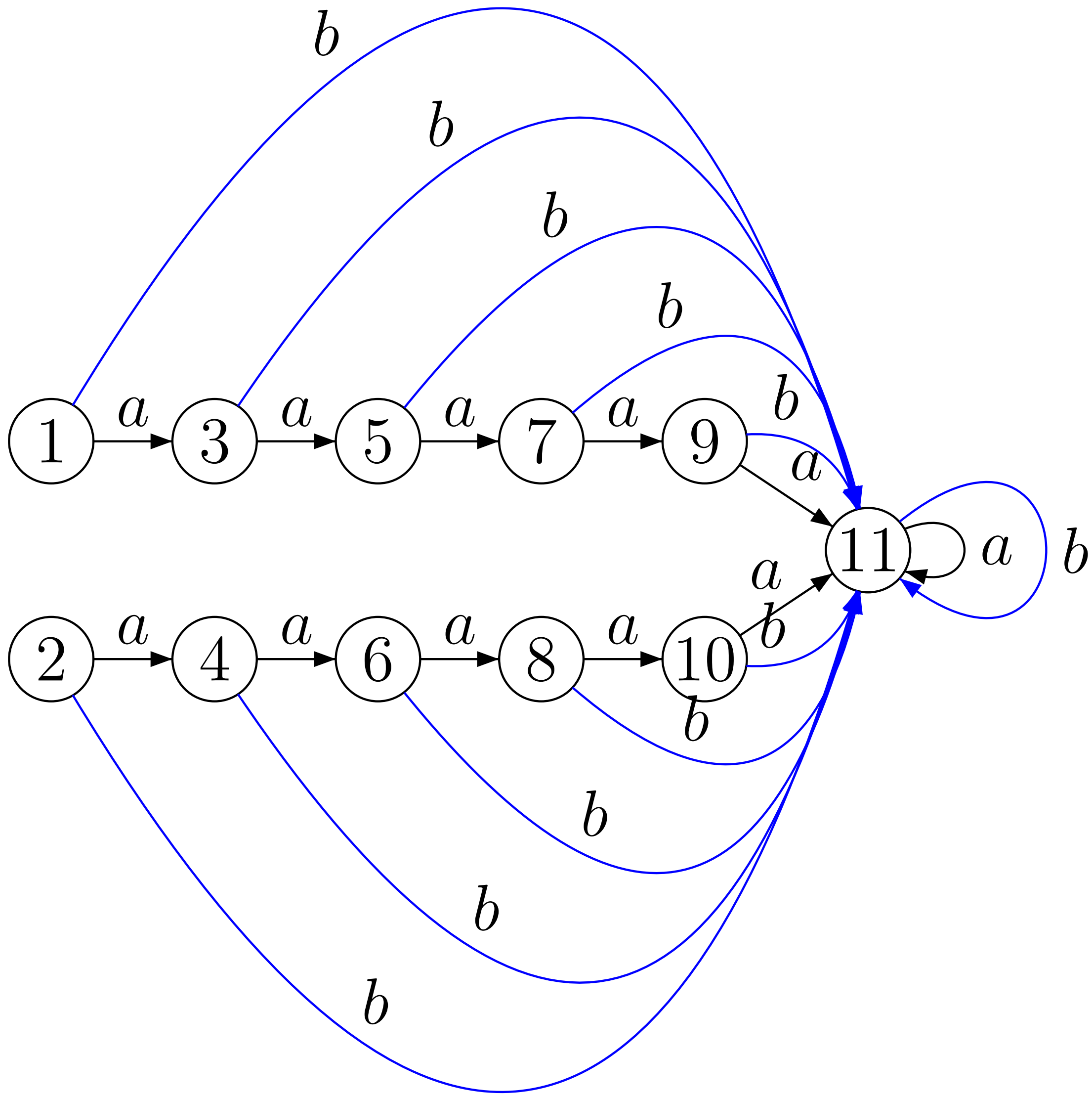
Proof For a given n consider $\mathscr{A}$ with $Q = \{0, ..., n-1\}$
$\Sigma = \{a_X | X \subseteq Q,\ |X| = k\} \cup \{a_Q\}$.
For $a_X \in \Sigma$ and $q \in Q$ let

$$\delta(q, a_X) = \begin{cases} \min(X), & if\ q \in X \\ q, & if\ q \notin X \end{cases}$$

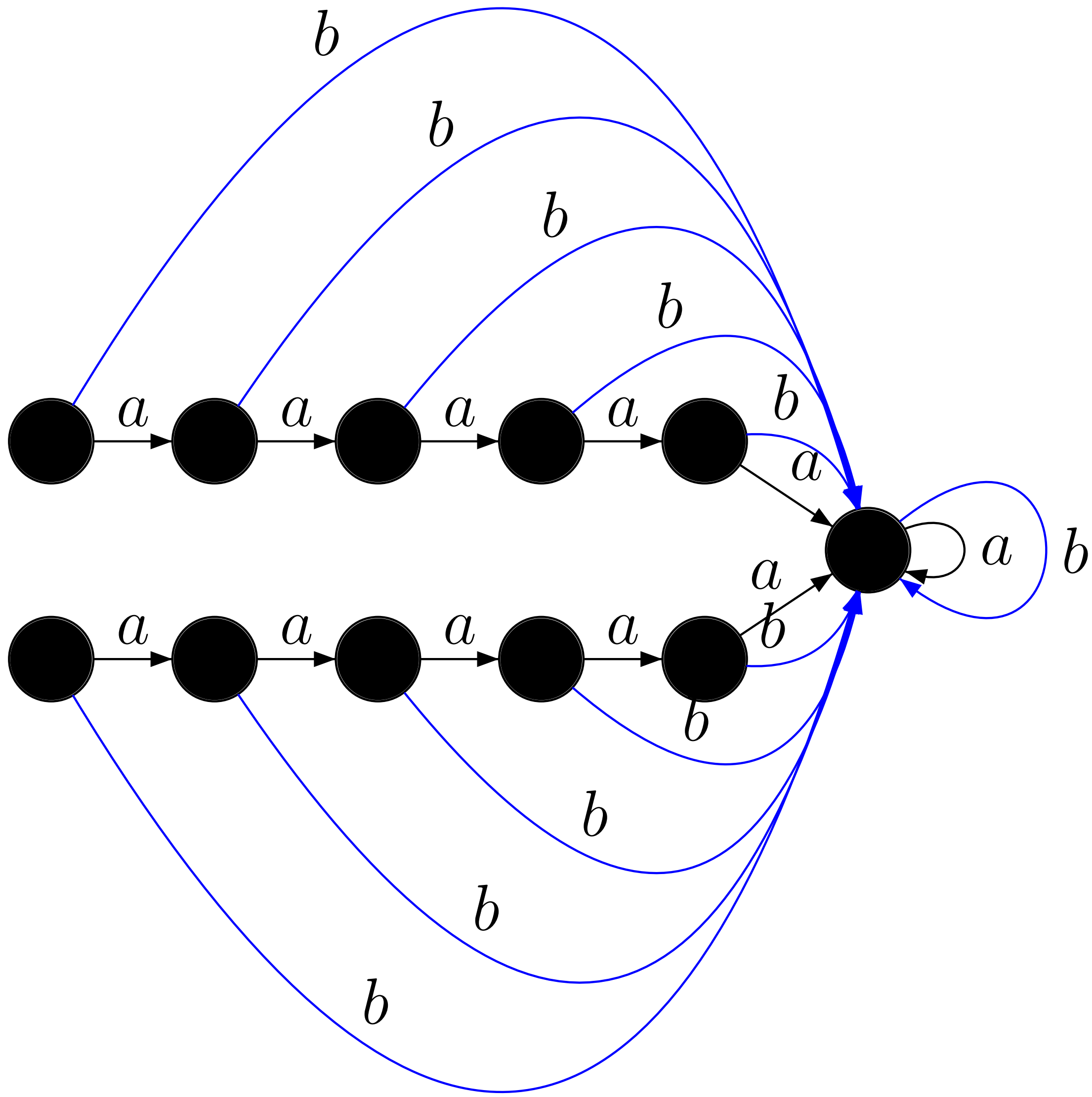
The shortest reset word is a_Q which has length 1.

If for each set P the algorithm chooses the word a_P instead of a_Q to merge the states, then only k states are merged at a time. In this case, the word u that the algorithm gives has length $\lceil \frac{n-1}{k-1} \rceil$.

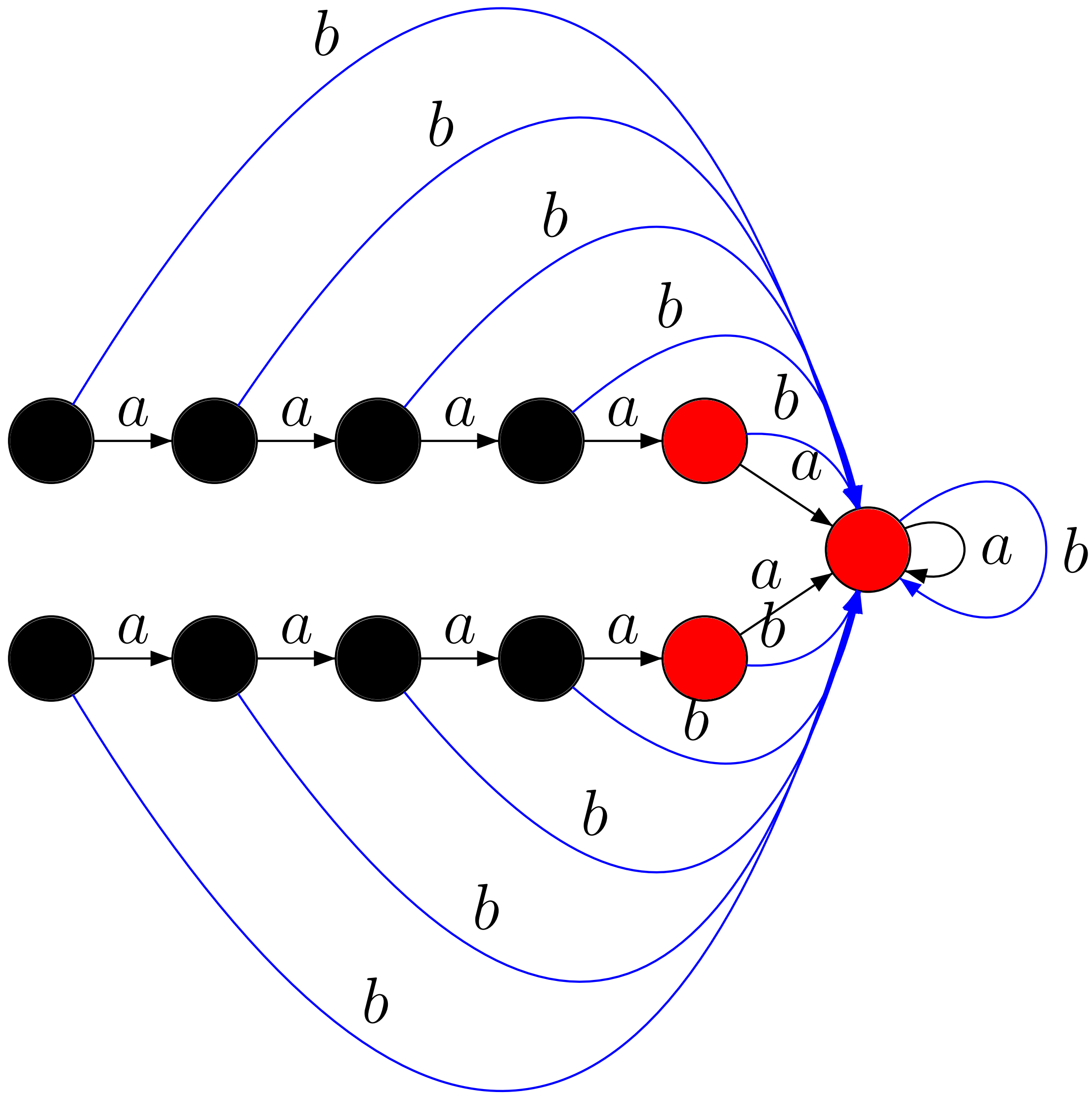
Example: 2 letters and greedy choices (k=3)



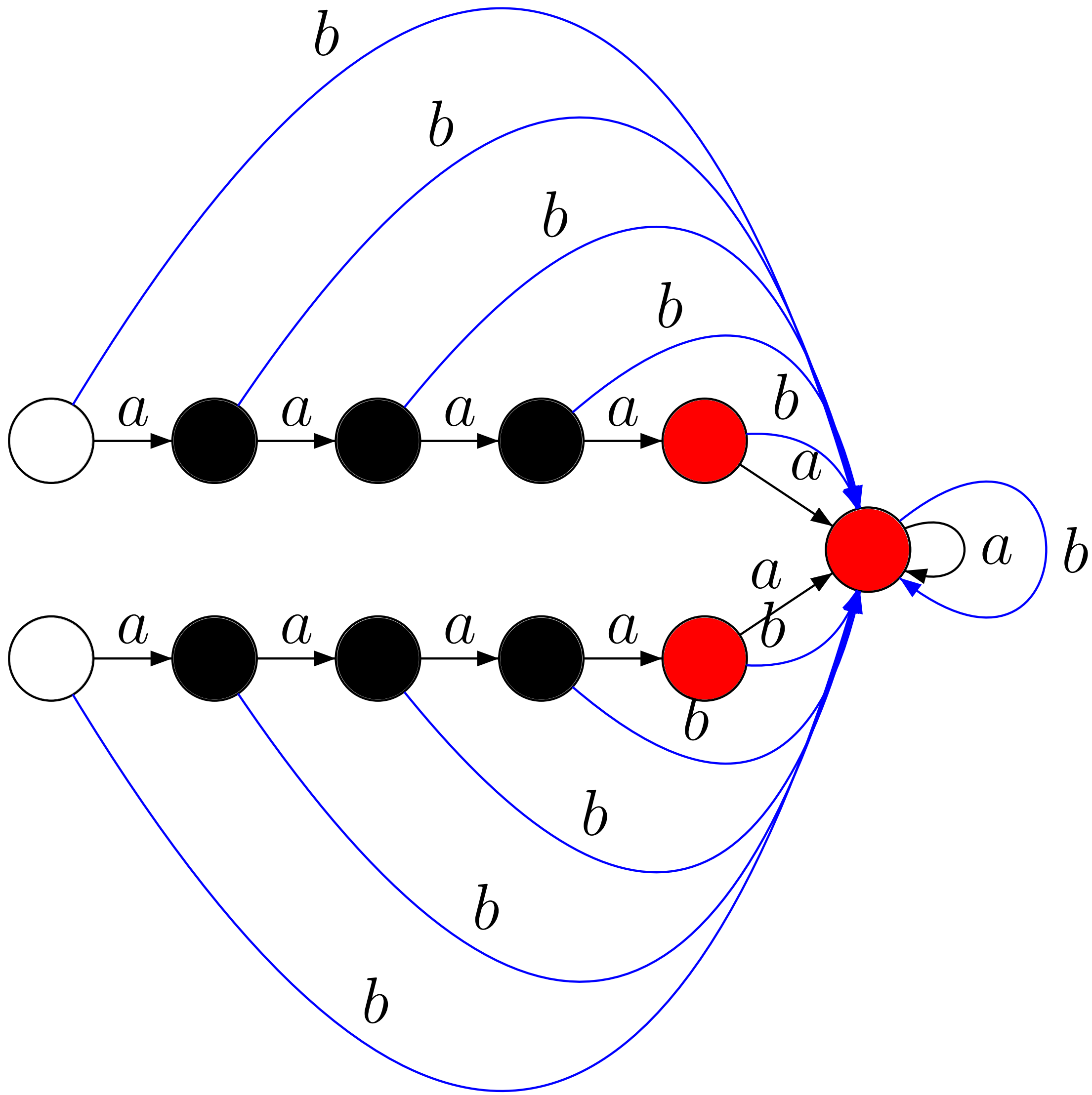
Example: 2 letters and greedy choices (k=3)



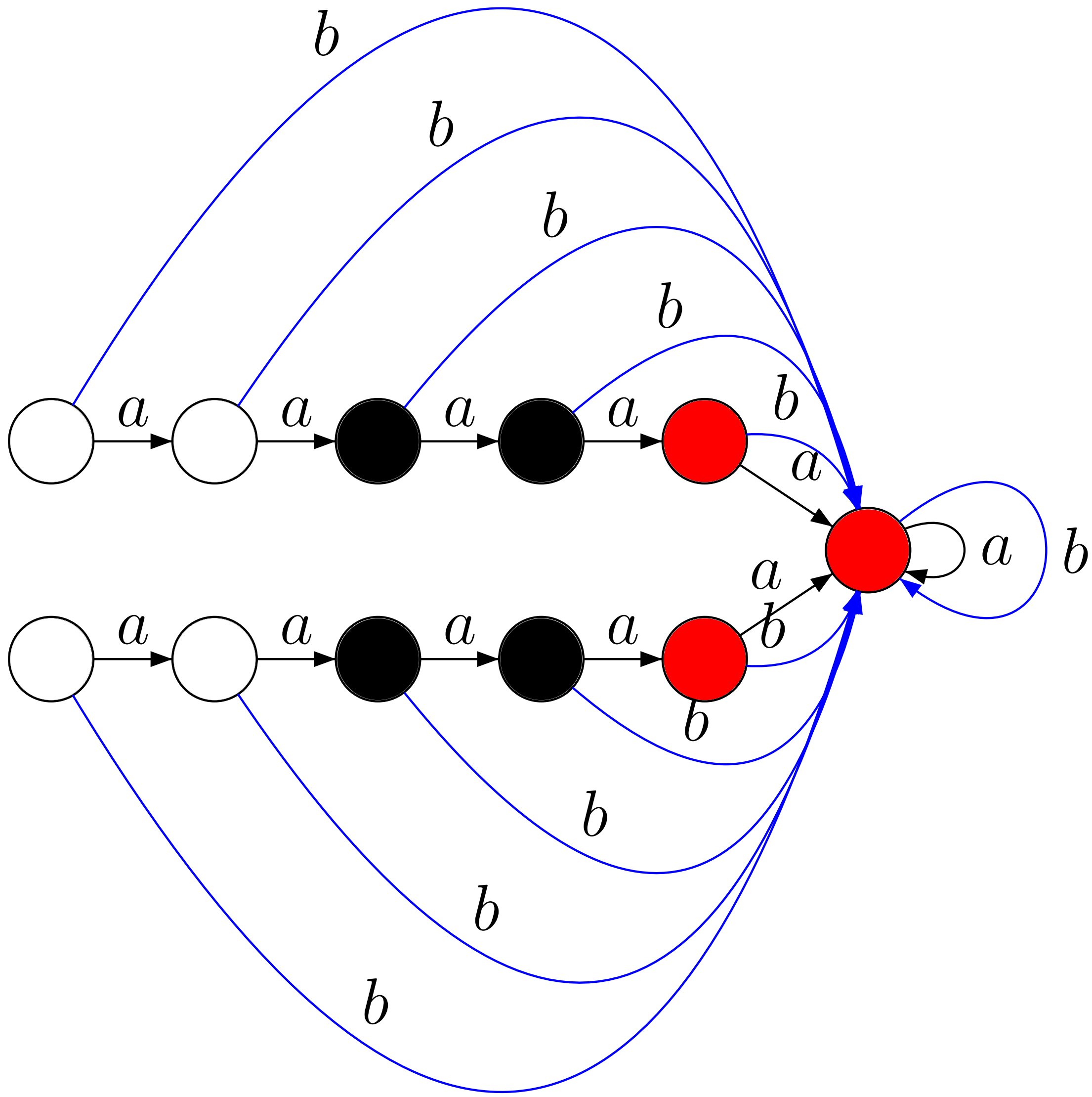
Example: 2 letters and greedy choices (k=3)



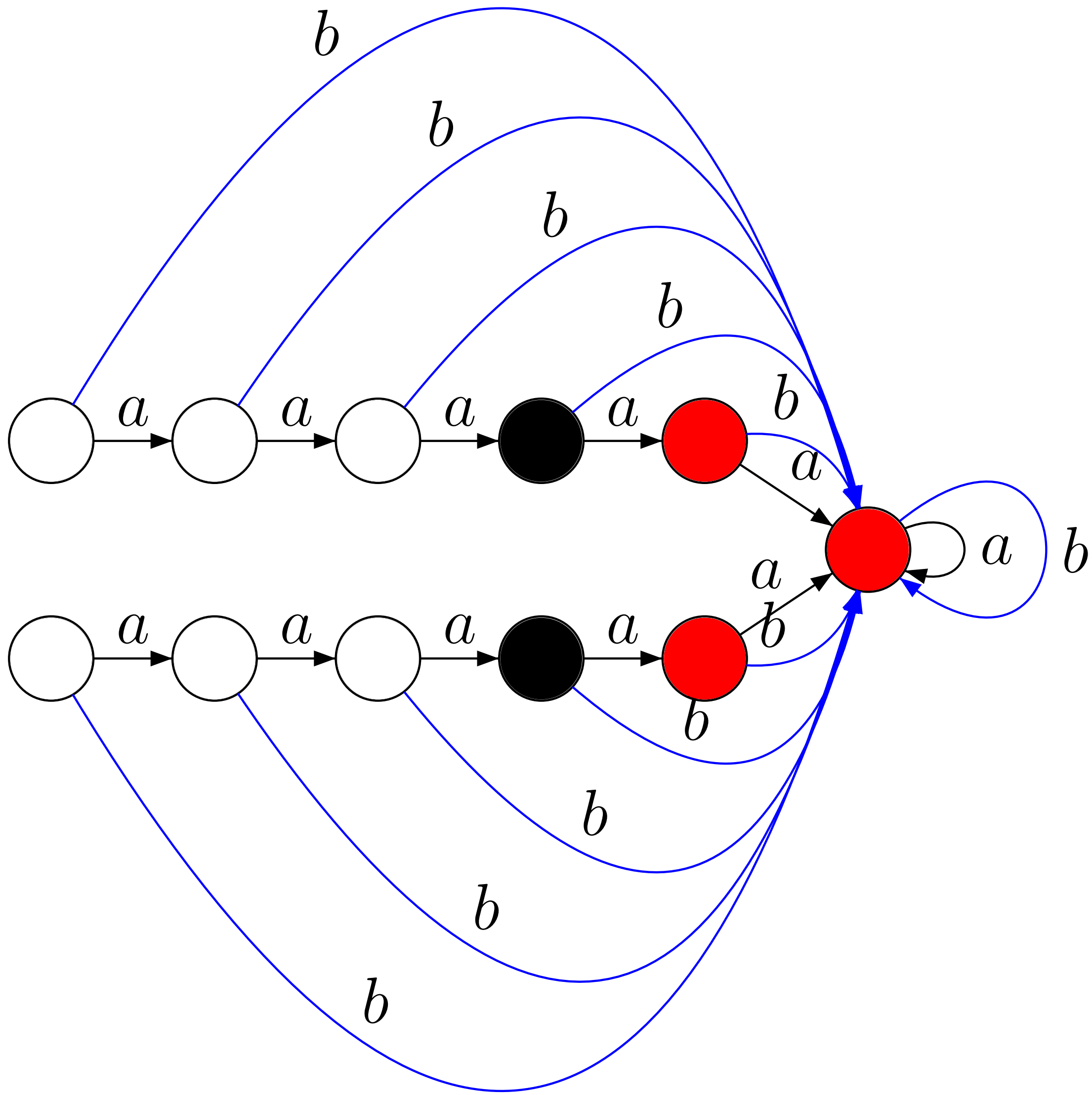
Example: 2 letters and greedy choices (k=3)



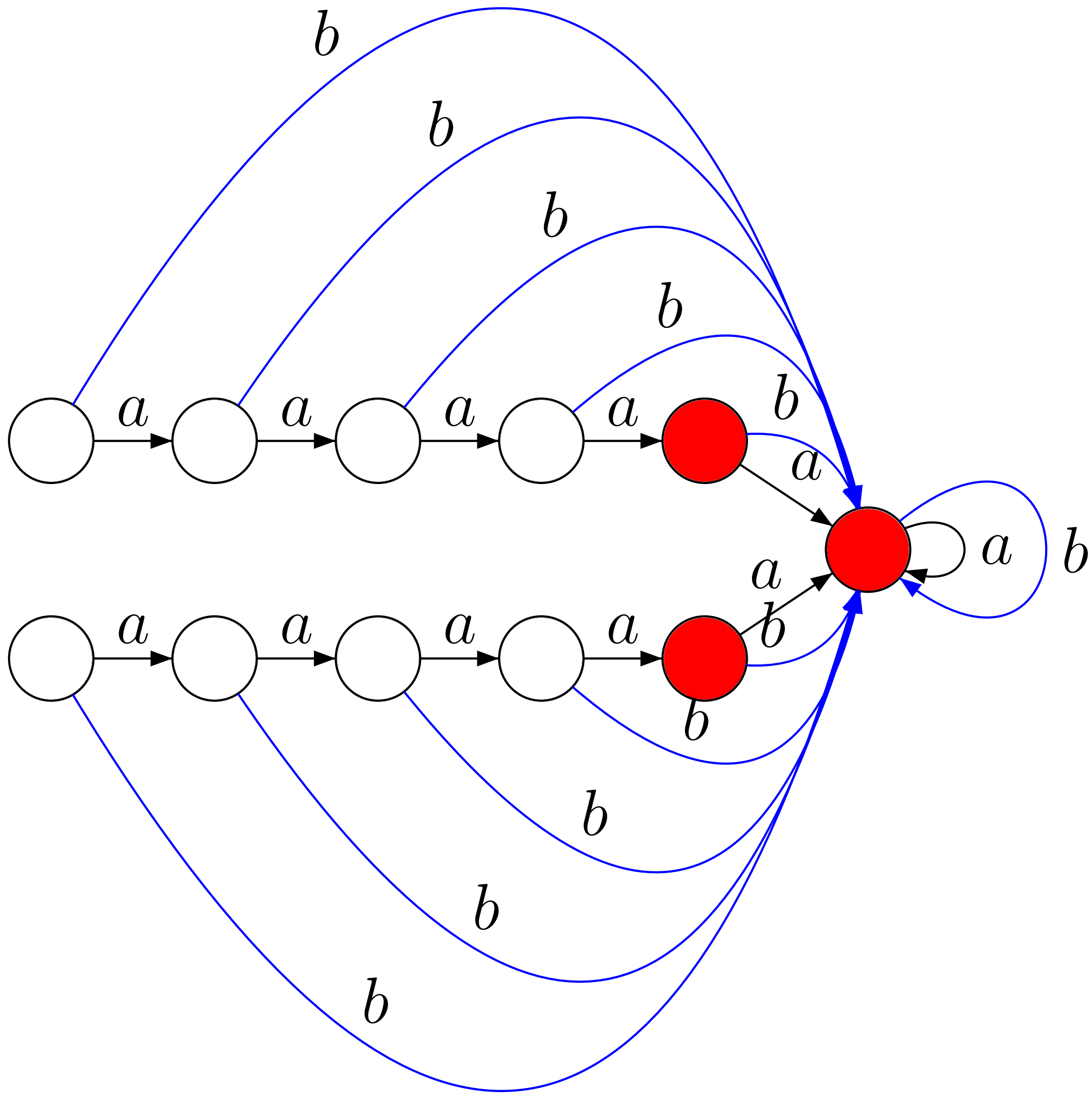
Example: 2 letters and greedy choices (k=3)



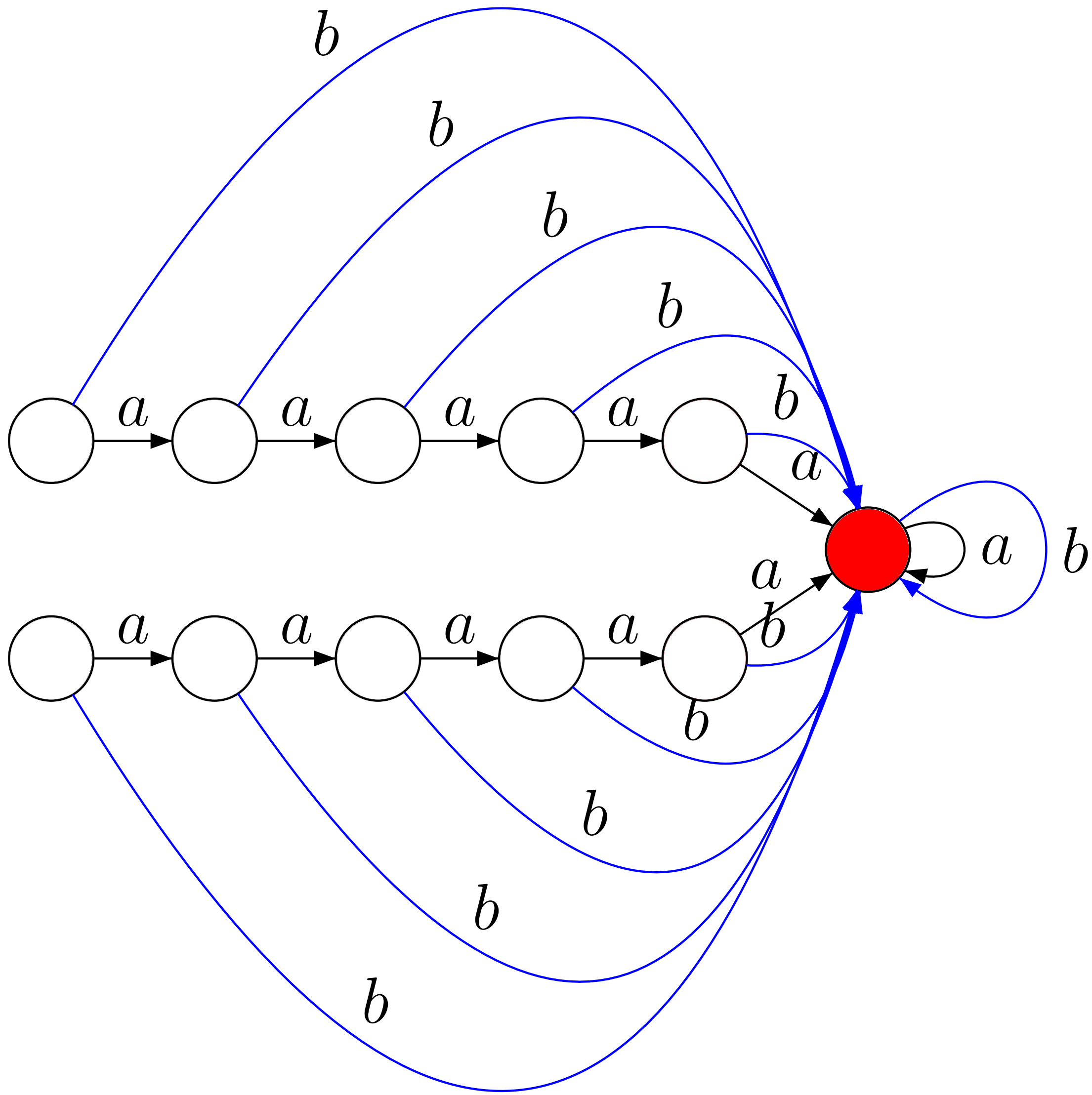
Example: 2 letters and greedy choices (k=3)



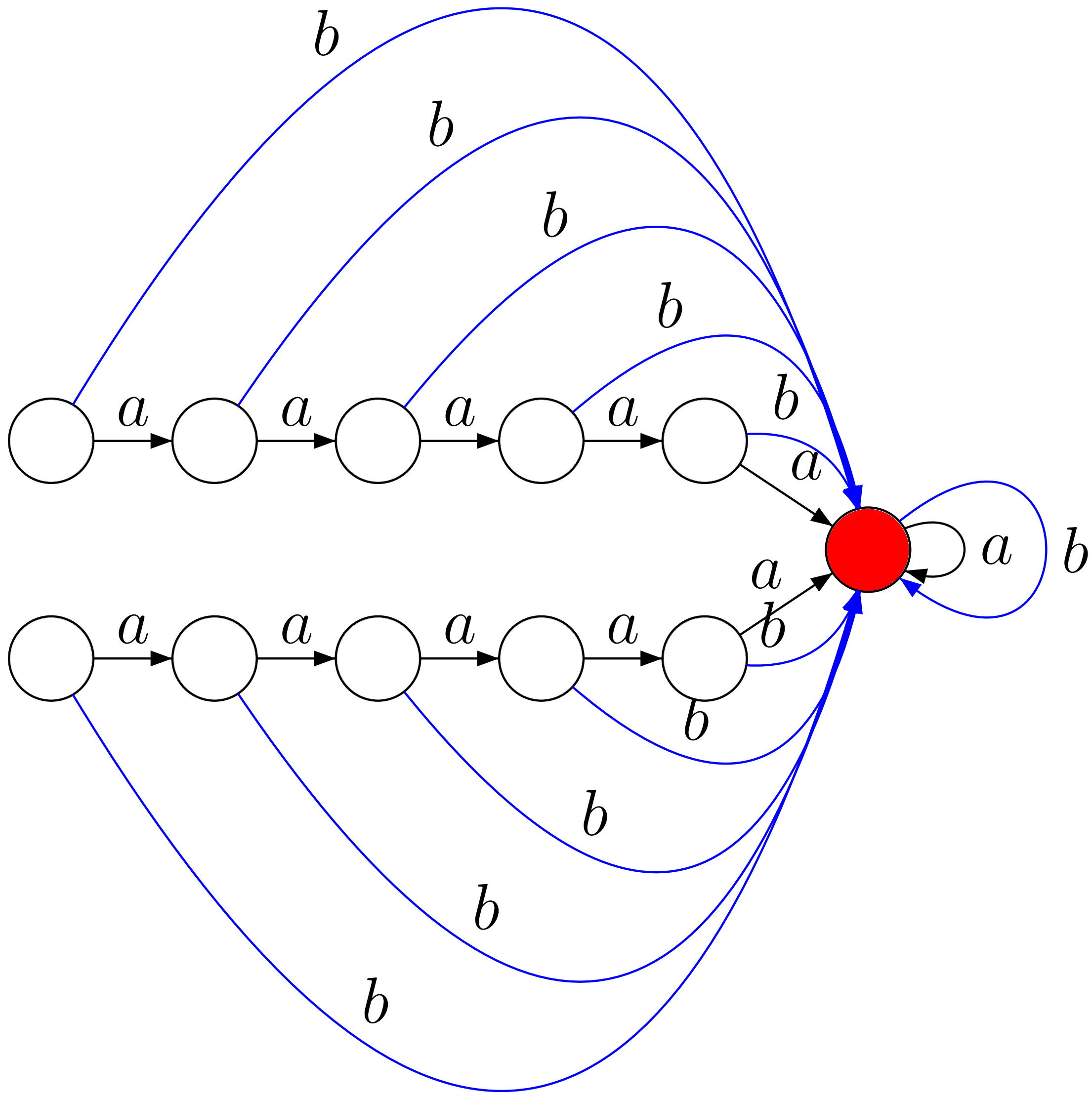
Example: 2 letters and greedy choices (k=3)



Example: 2 letters and greedy choices (k=3)



Example: 2 letters and greedy choices (k=3)



approximation ratio 5 or $\lceil \frac{n-1}{k-1} \rceil$

Perfect greedy algorithm

Let the algorithm selects the best set P and the best of the shortest words and the in terms of minimizing the length of the result.

Perfect greedy algorithm

Let the algorithm selects the best set P and the best of the shortest words and the in terms of minimizing the length of the result.

We call such an algorithm a *perfect greedy*.

Perfect greedy algorithm

Let the algorithm selects the best set P and the best of the shortest words and the in terms of minimizing the length of the result.

We call such an algorithm a *perfect greedy*.

Perfect greedy choices may require over polynomial-time computation.

Perfect greedy algorithm

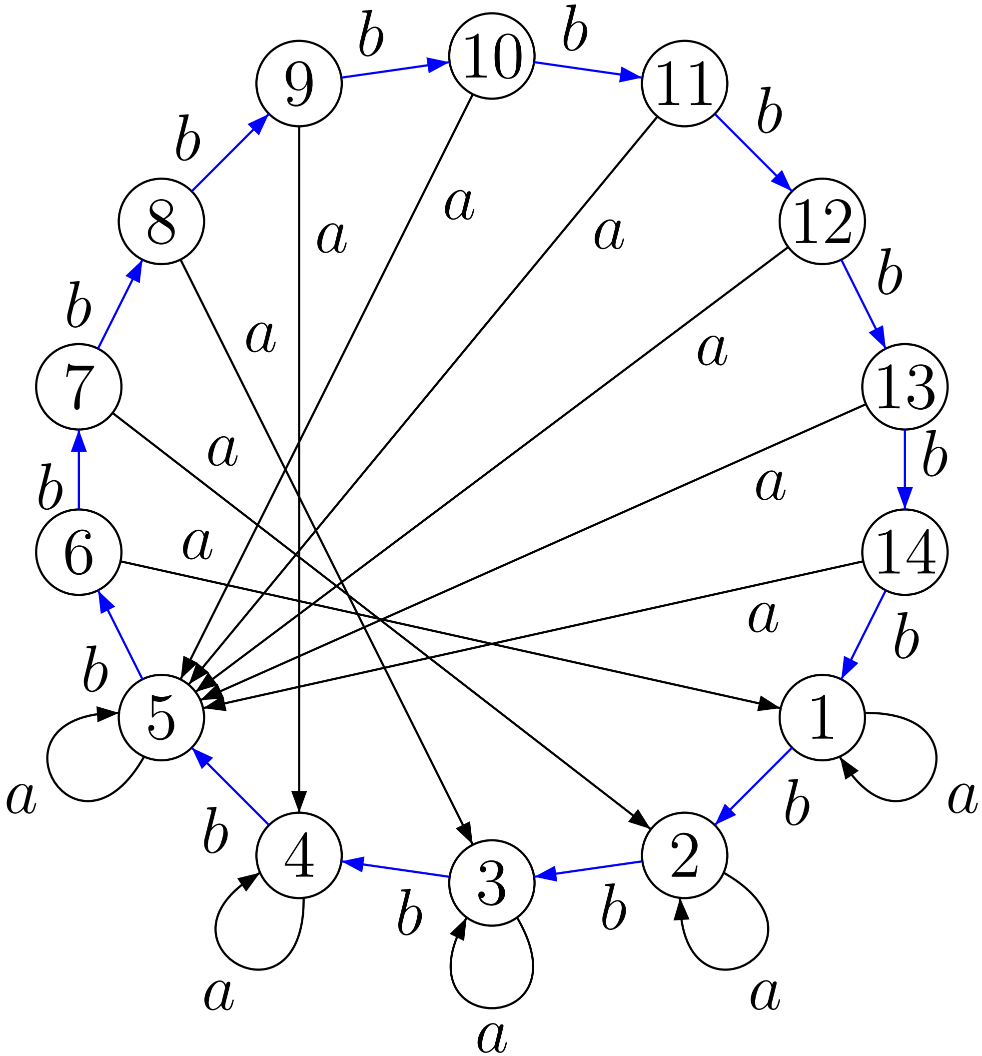
Let the algorithm selects the best set P and the best of the shortest words and the in terms of minimizing the length of the result.

We call such an algorithm a *perfect greedy*.

Perfect greedy choices may require over polynomial-time computation.

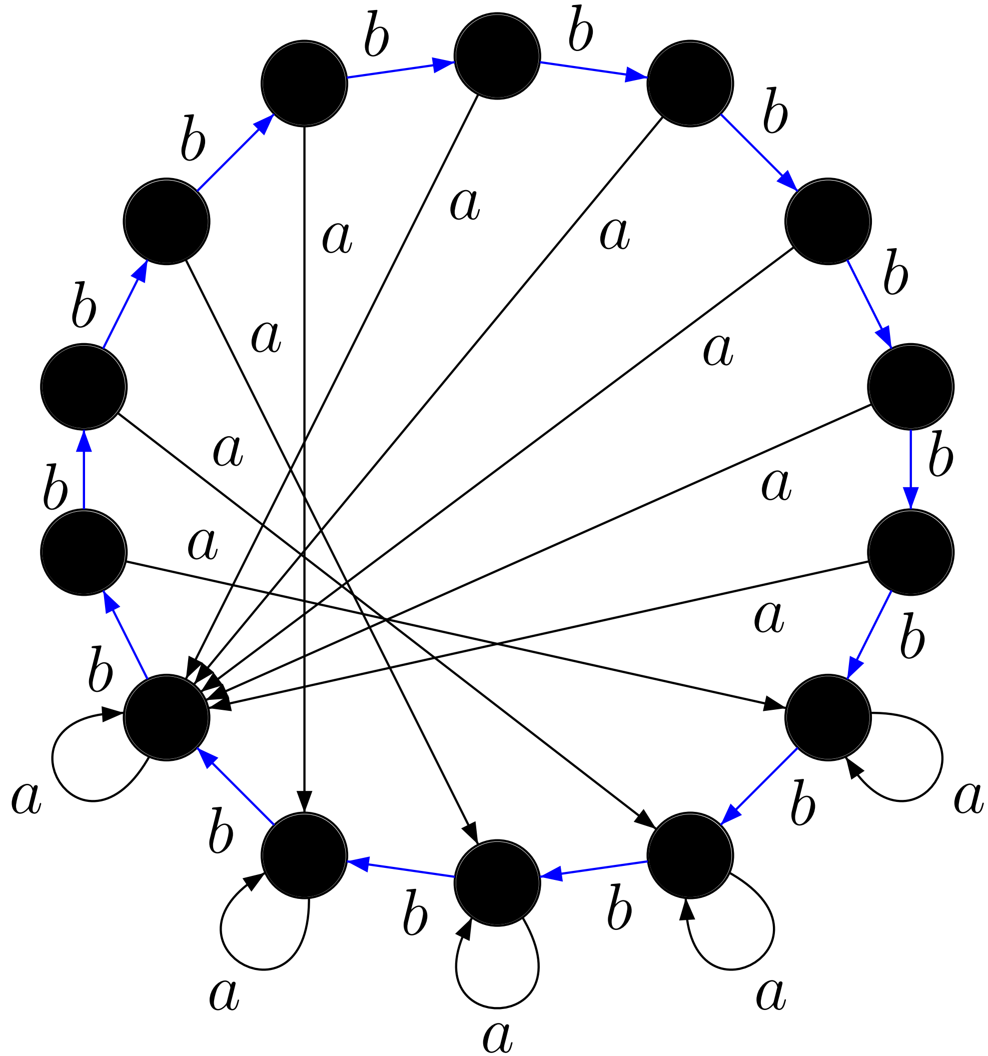
The example for perfect greedy algorithm gives the lower bound of approximation ratio for arbitrary greedy algorithm

“Honest” example: shortest reset word



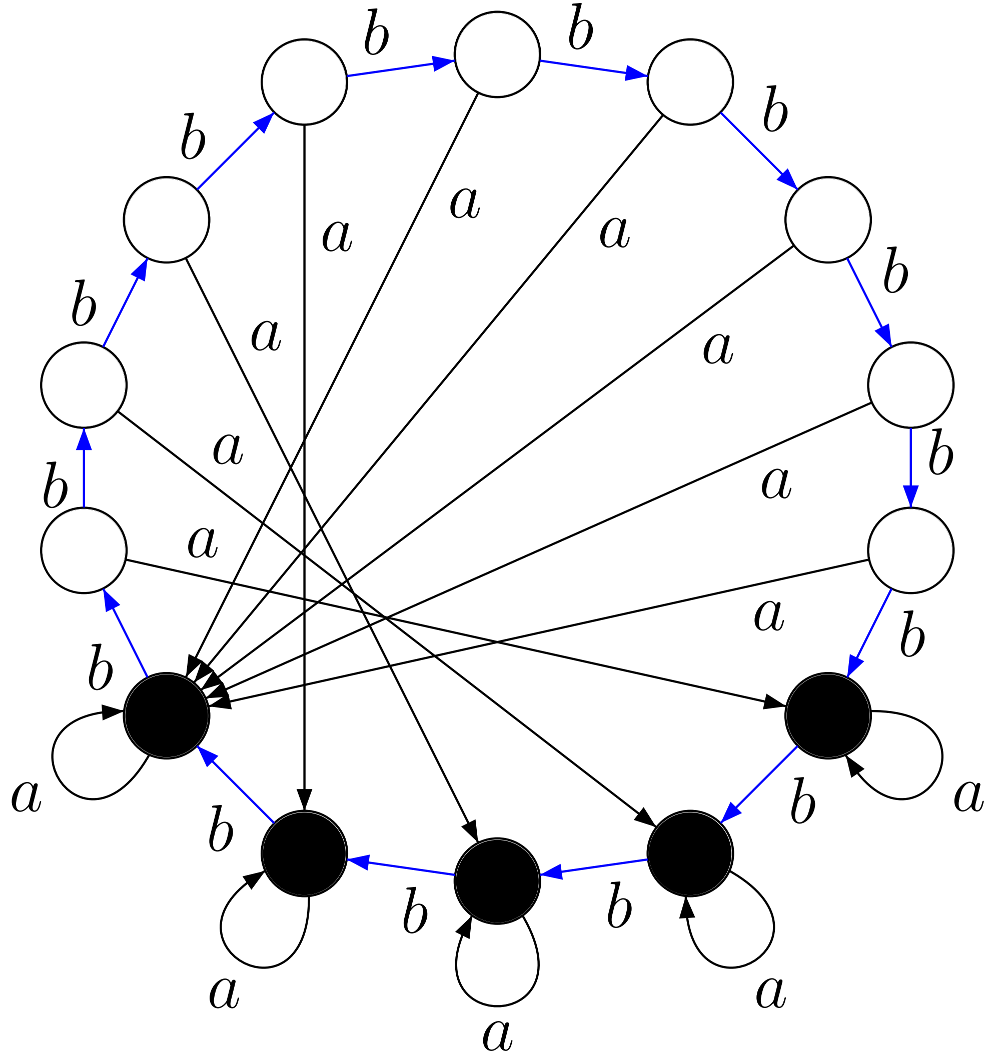
$abbbbbbbba$

“Honest” example: shortest reset word



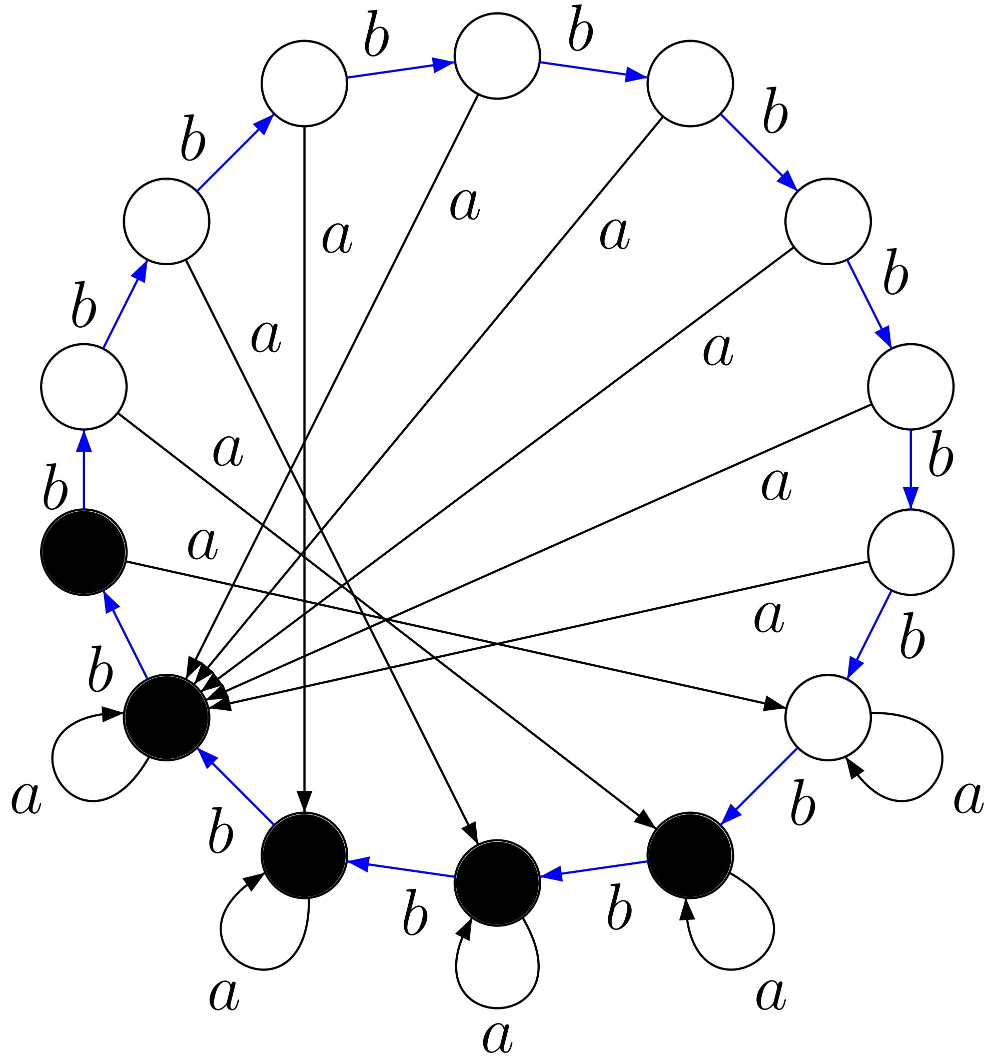
$abbbbbbbba$

“Honest” example: shortest reset word



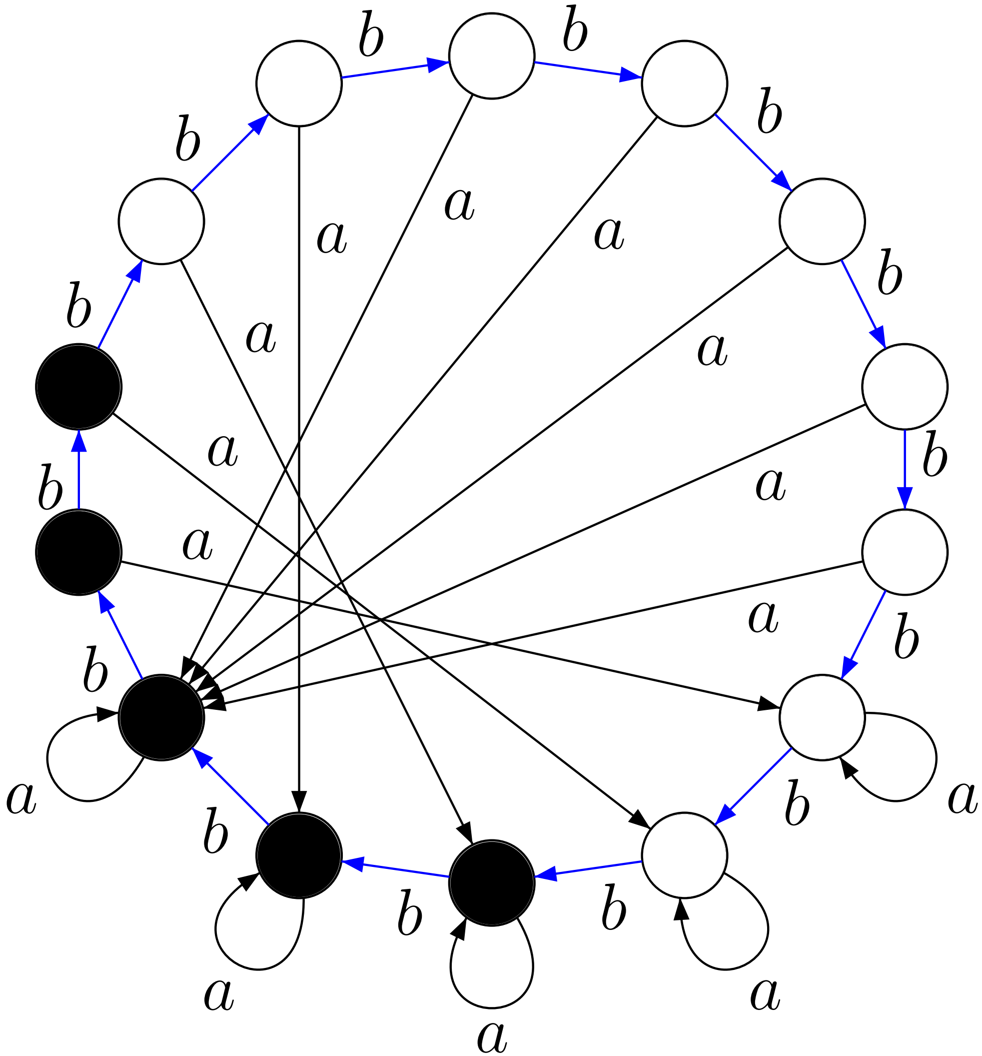
$abbbbbbbba$

“Honest” example: shortest reset word



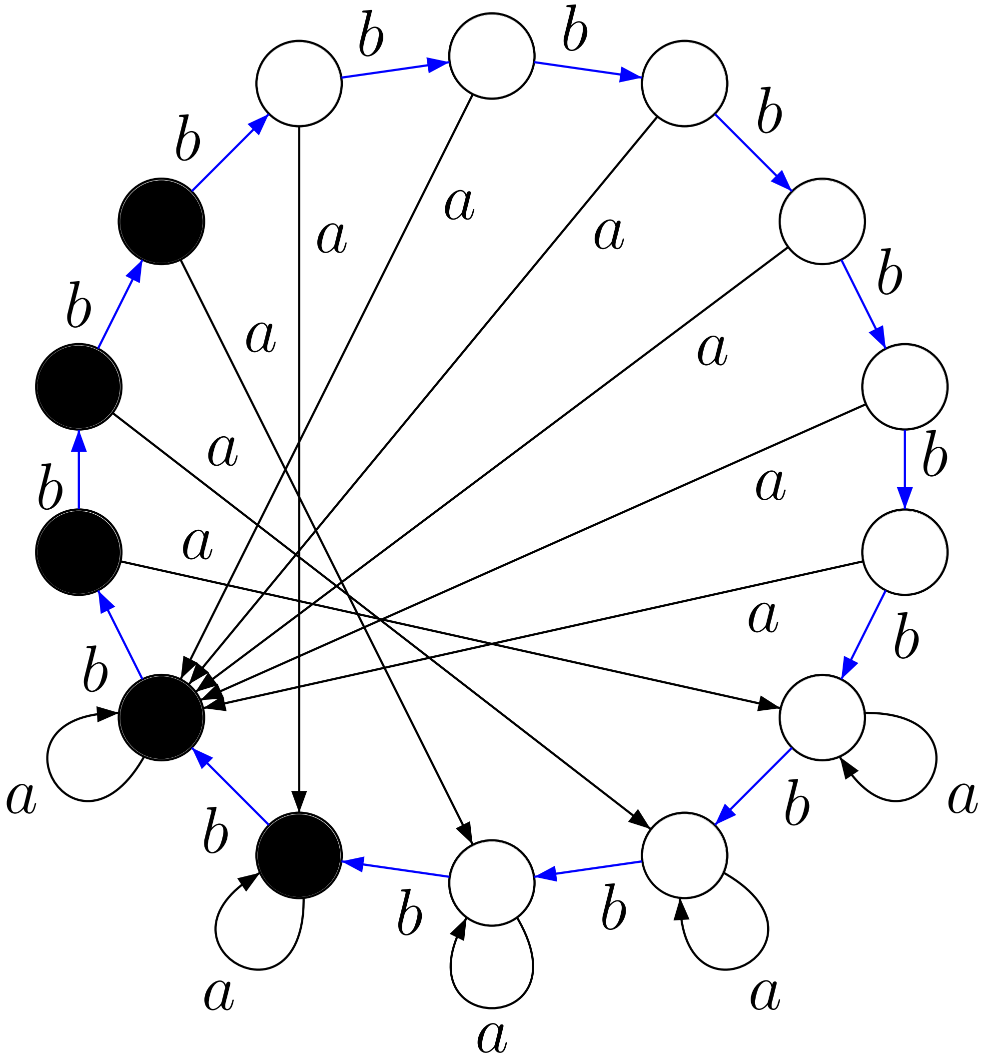
$abbbbbbbba$

“Honest” example: shortest reset word



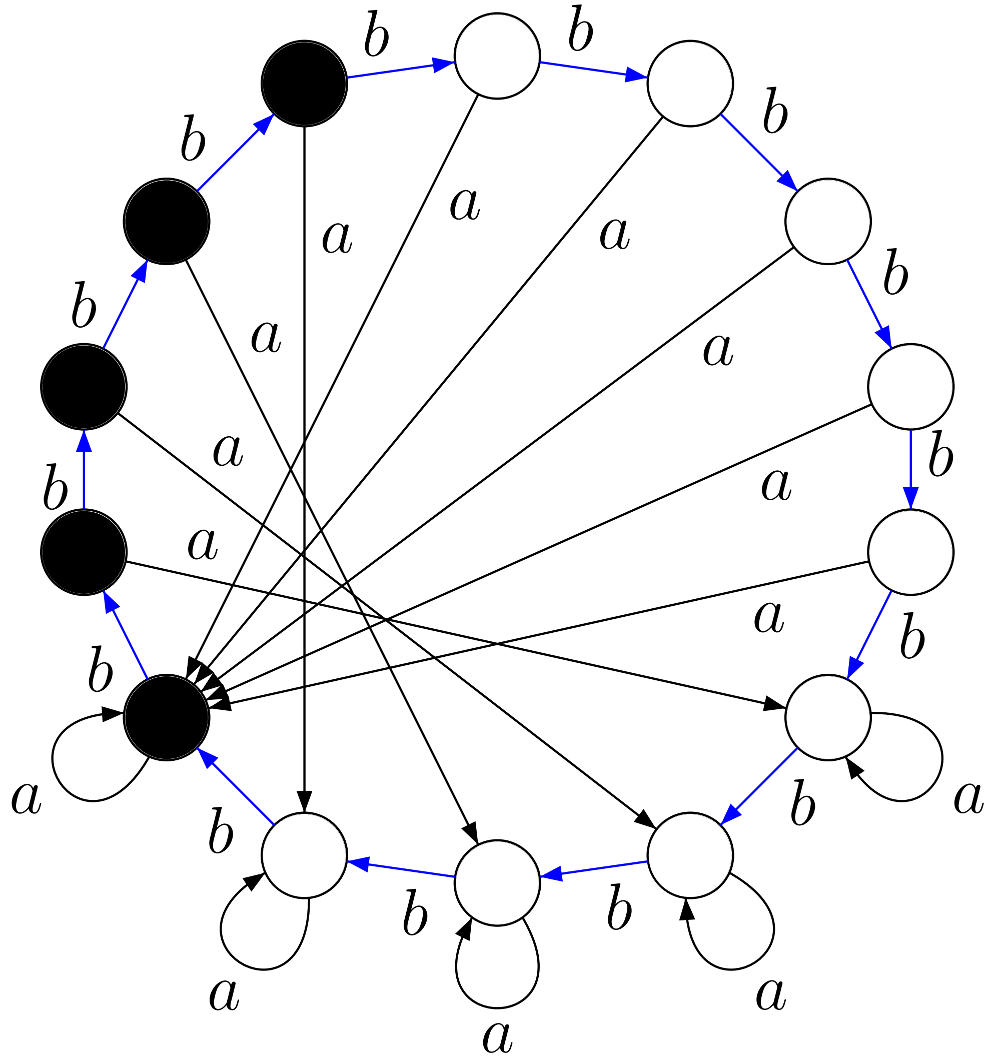
$ab\textcolor{red}{b}bbbbbbba$

“Honest” example: shortest reset word



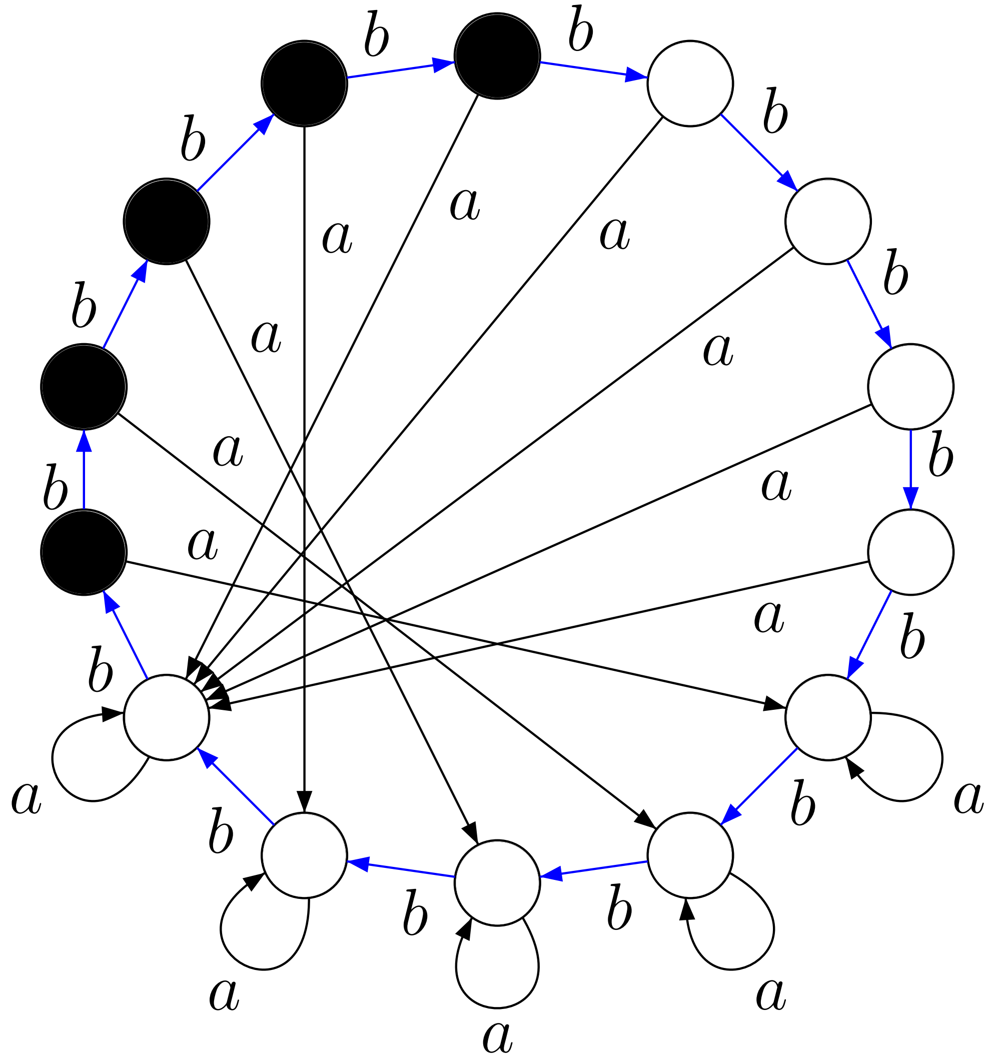
$ab\color{red}{b}bbbbbba$

"Honest" example: shortest reset word



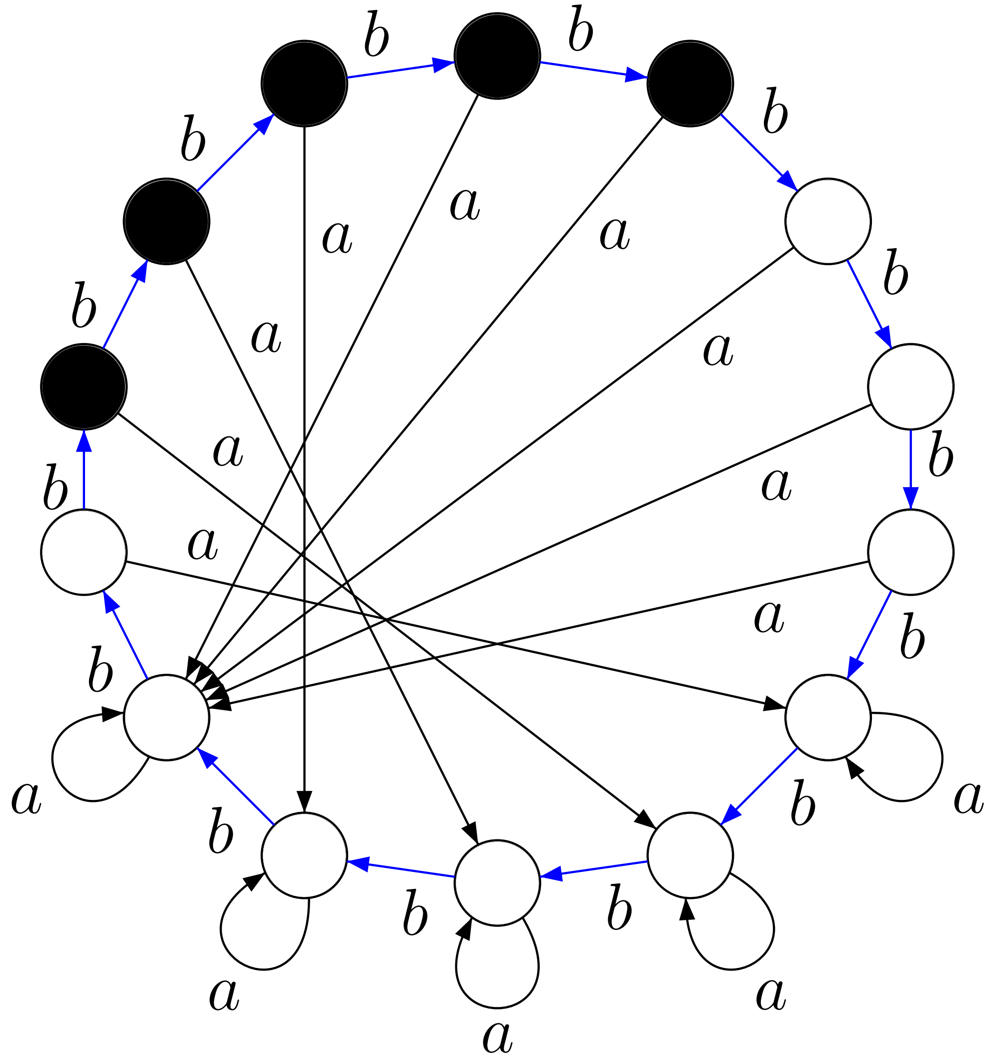
$abbbbbbbba$

“Honest” example: shortest reset word



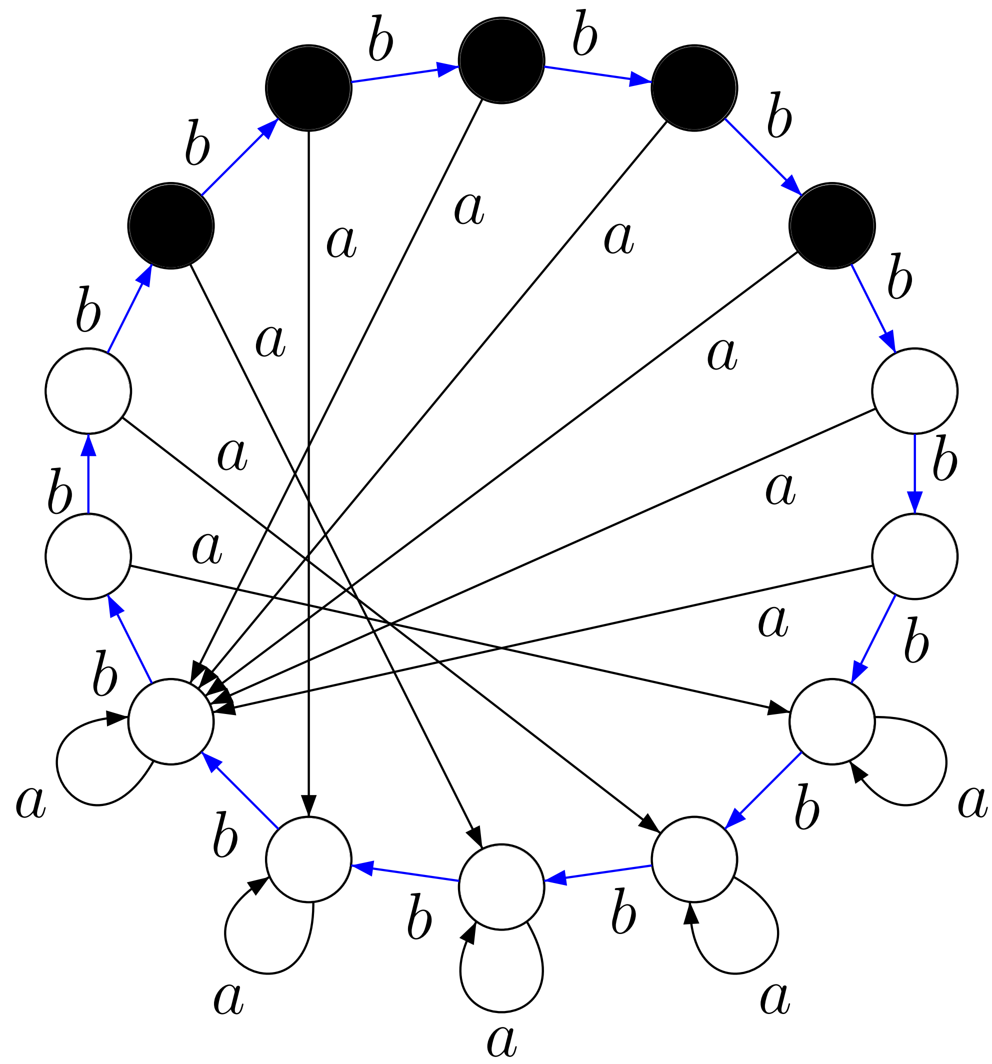
$abbbb\color{red}{b}\color{black}{bbbba}$

“Honest” example: shortest reset word



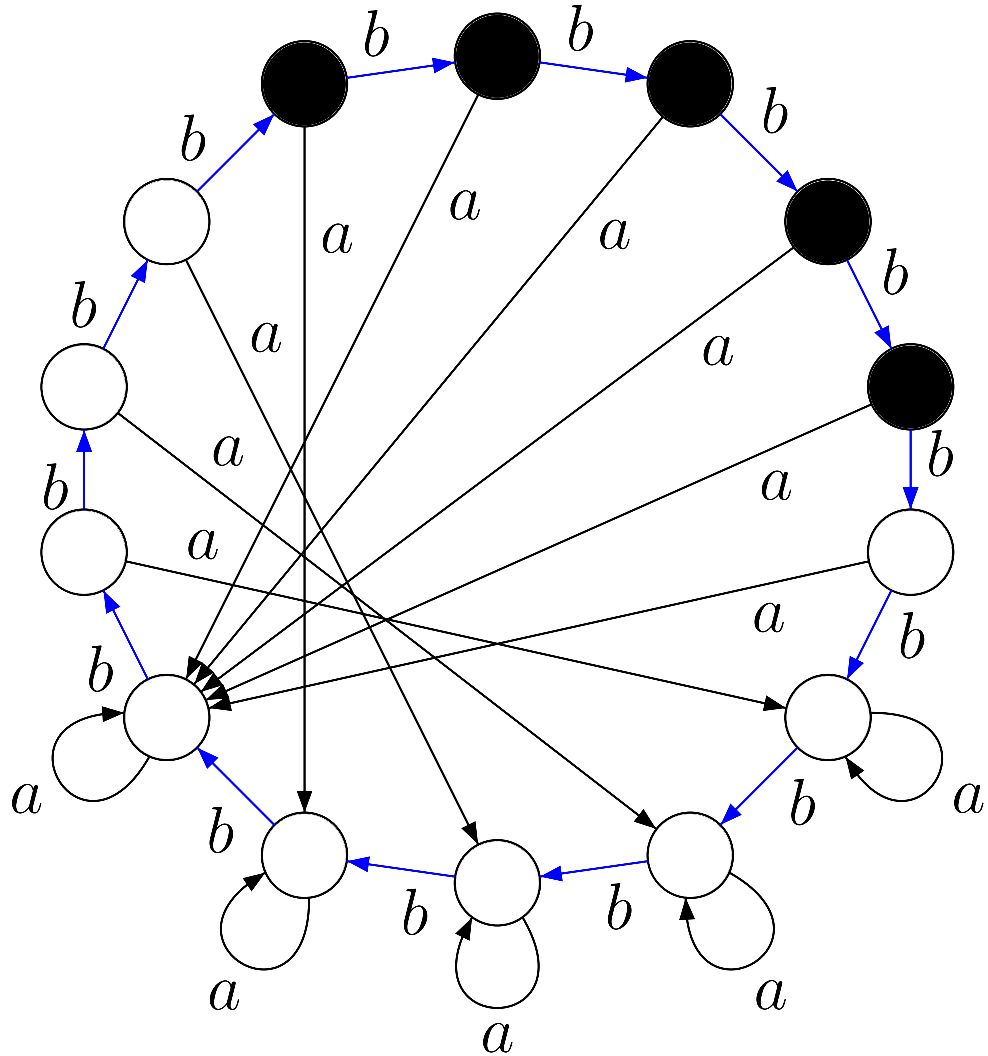
$abbbbbbbbba$

“Honest” example: shortest reset word



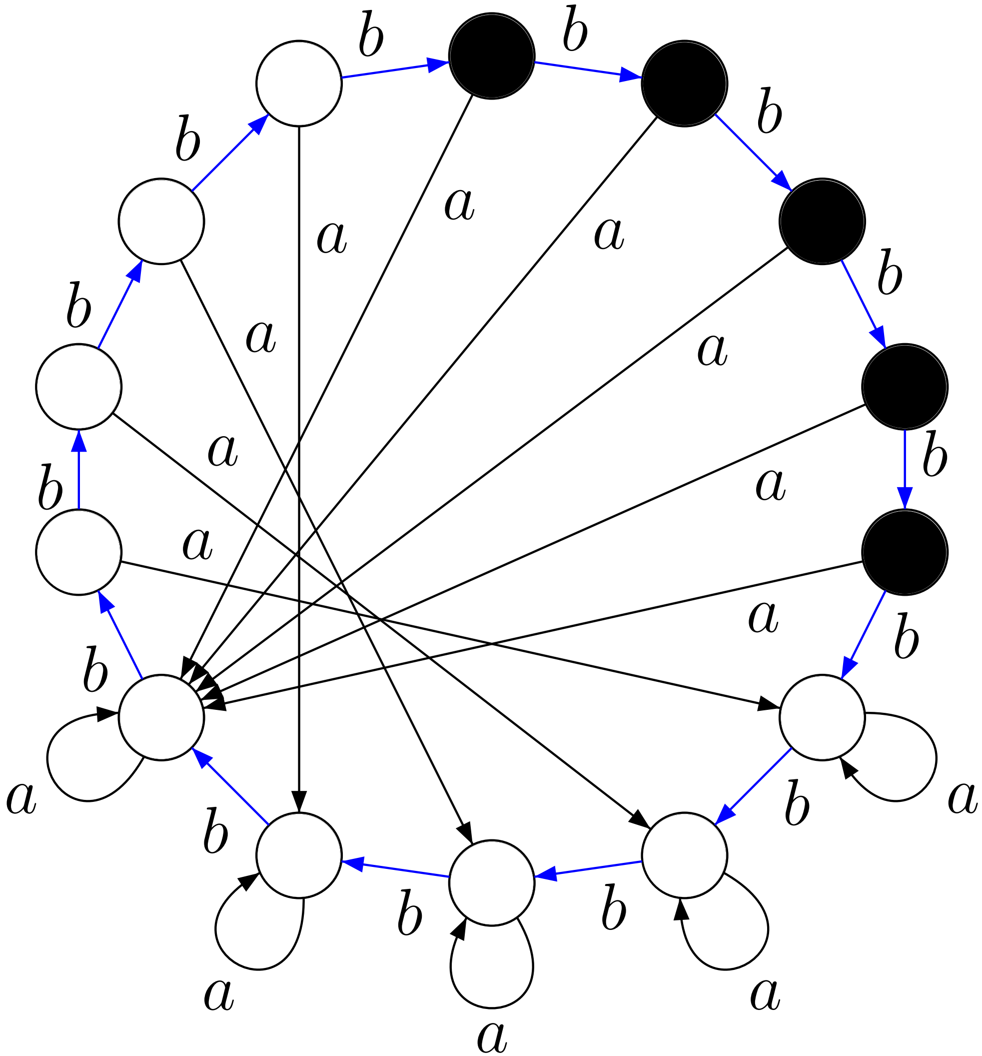
$abbbbbb\textcolor{red}{b}bba$

“Honest” example: shortest reset word



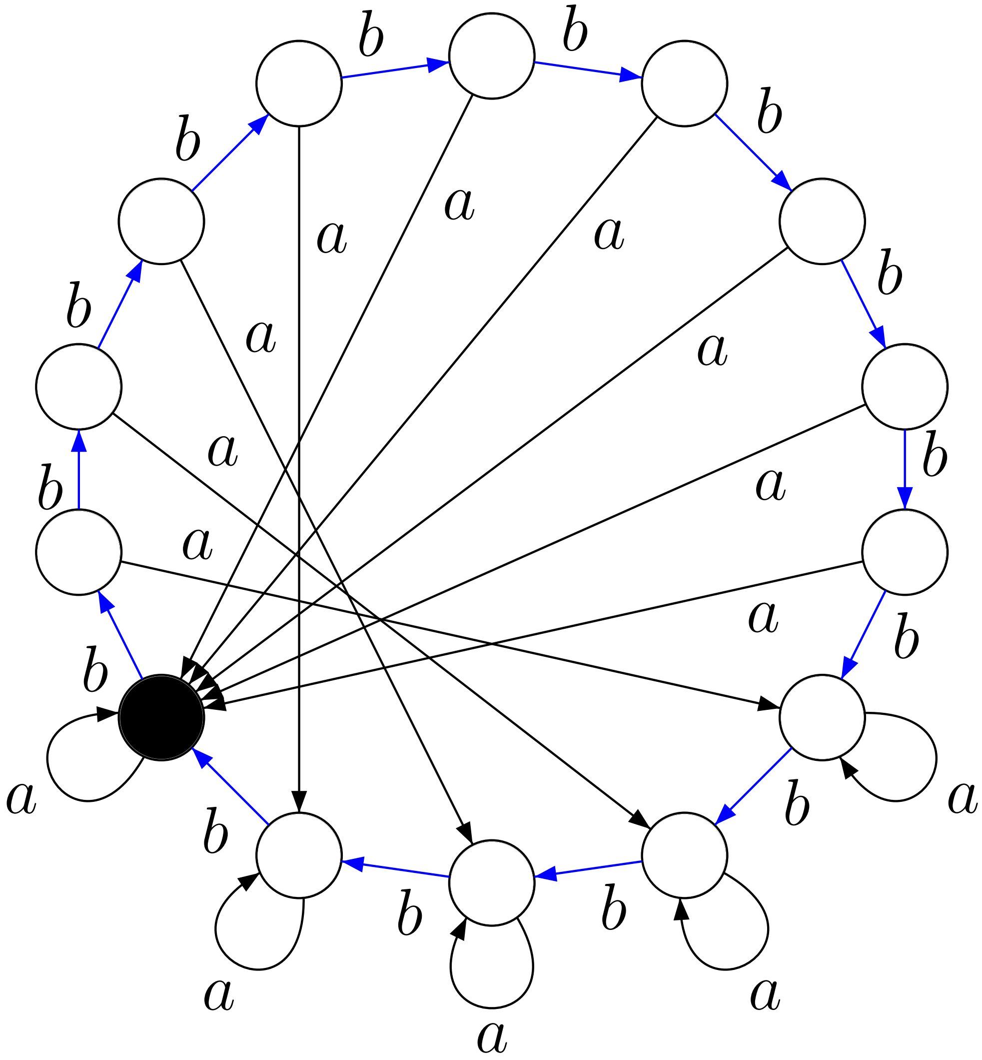
$abbbbbbbbba$

“Honest” example: shortest reset word



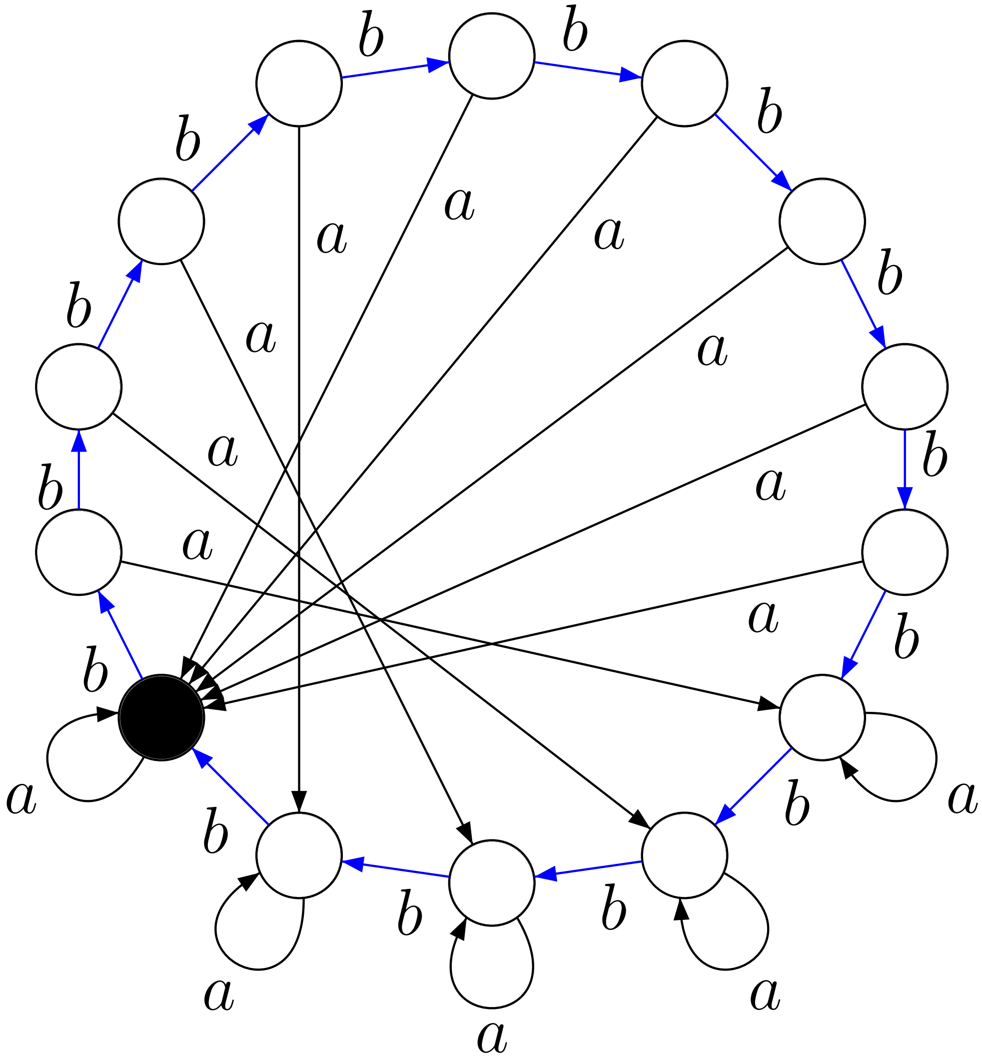
$abbbbbbbba$

"Honest" example: shortest reset word



$abbbbbbbba$

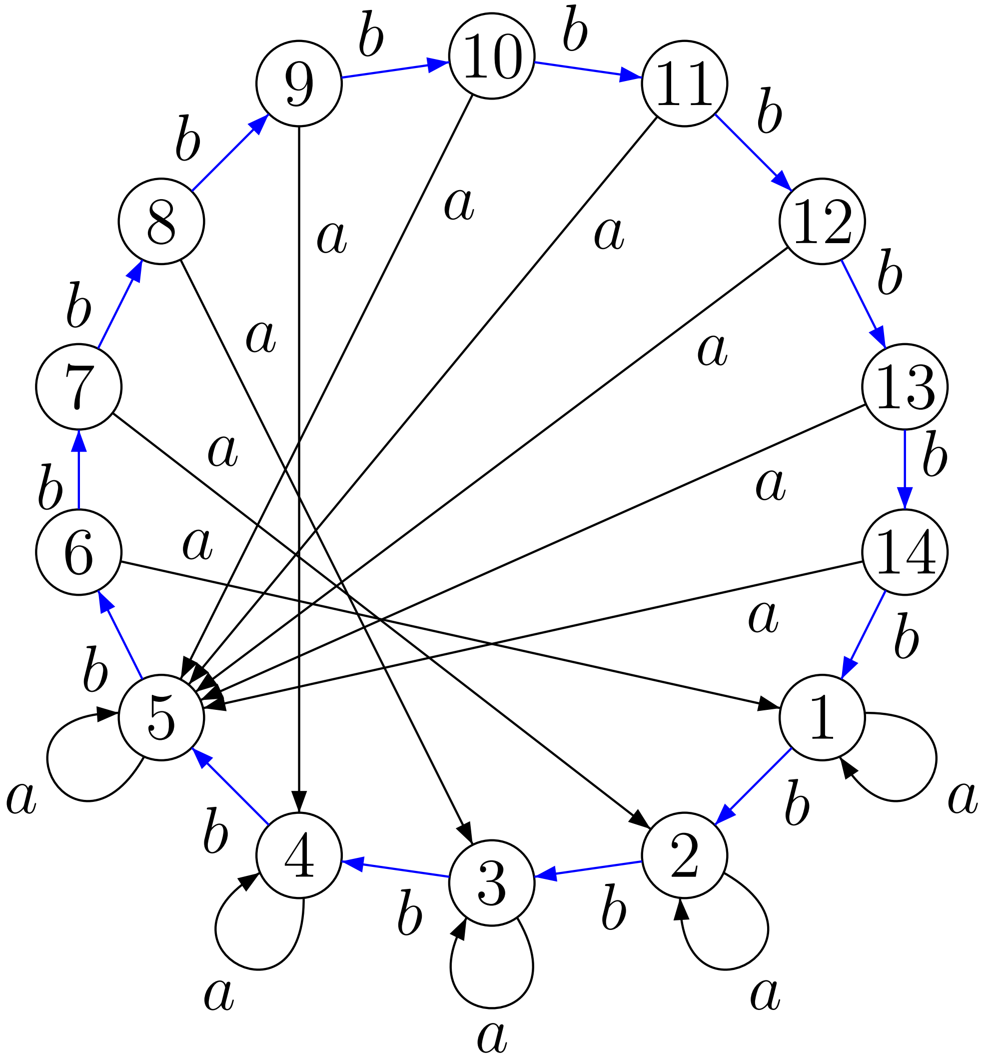
“Honest” example: shortest reset word



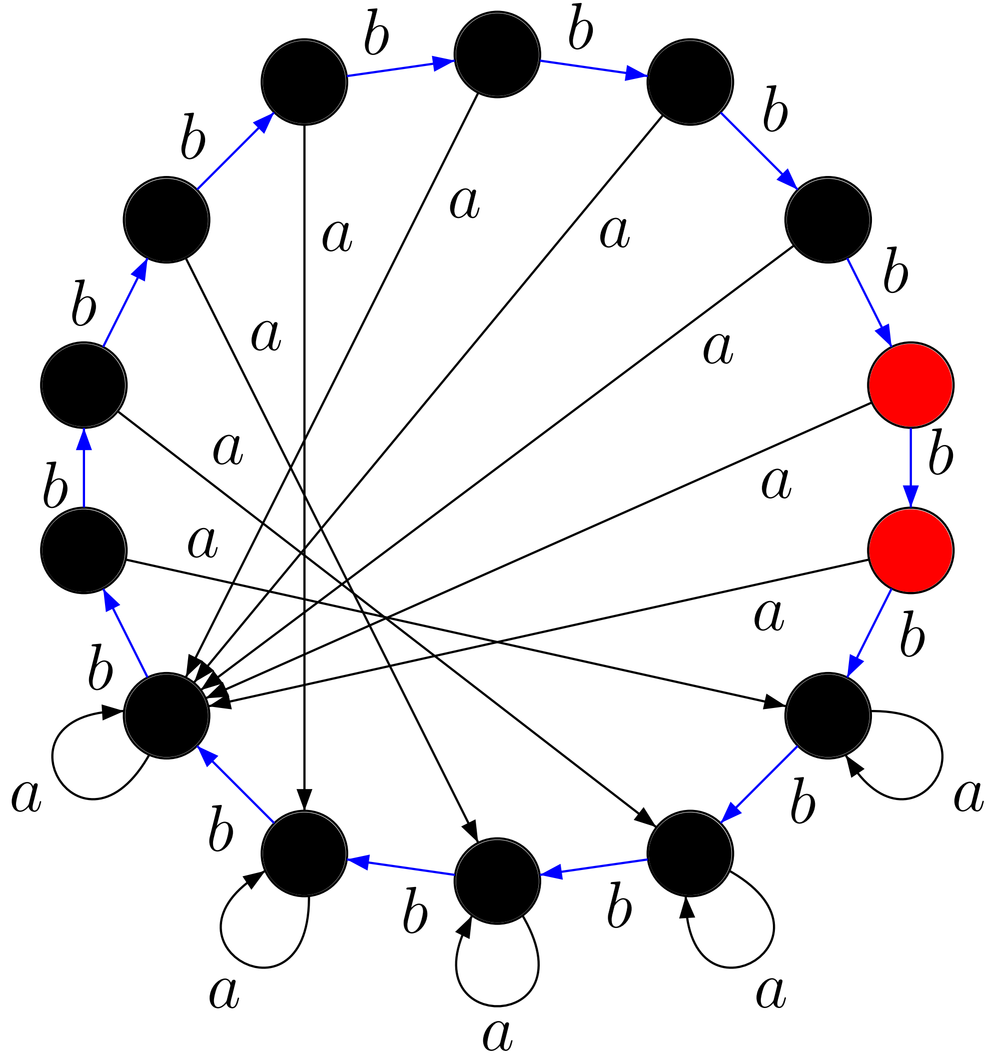
$ab b b b b b b b b \textcolor{red}{a}$

reset threshold 11 or $\frac{2(n+1)}{3} + 1$

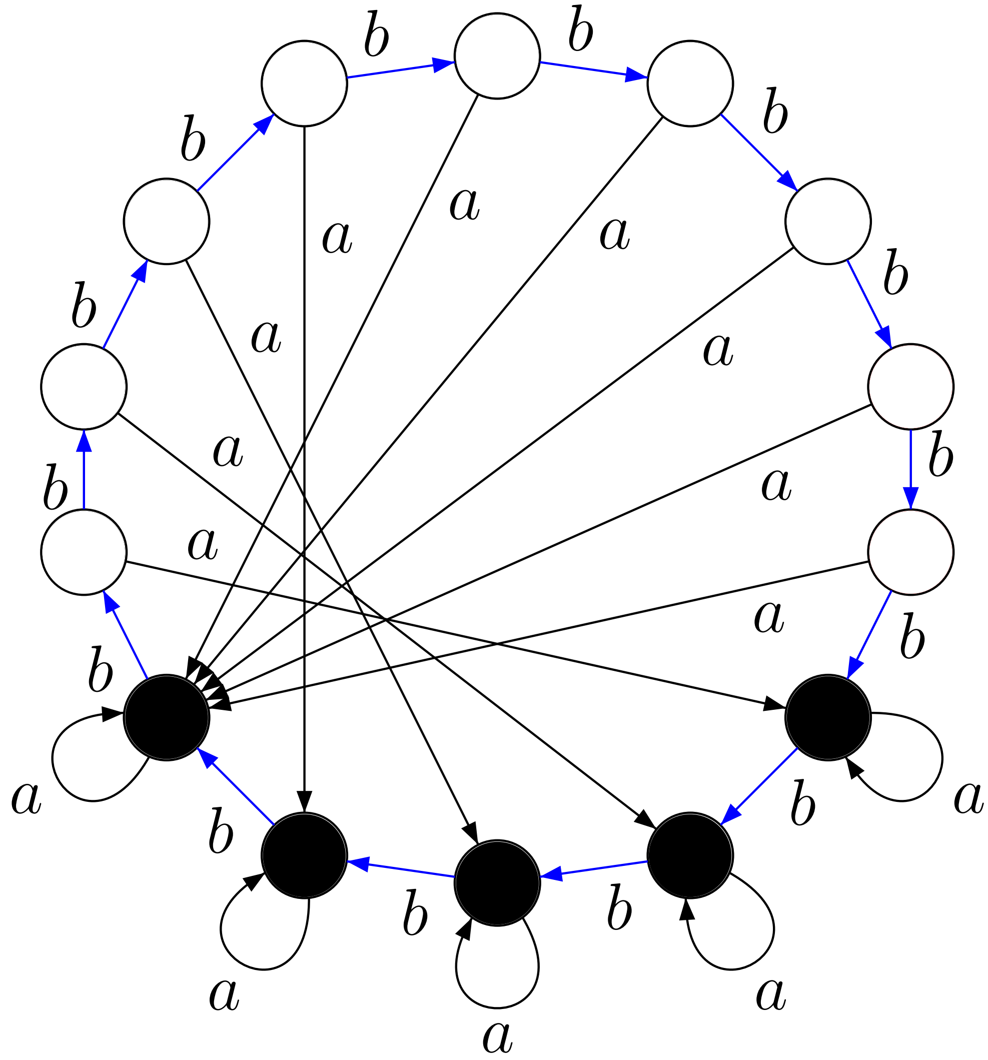
“Honest” example: “greedy” reset word (k=2)



“Honest” example: “greedy” reset word (k=2)

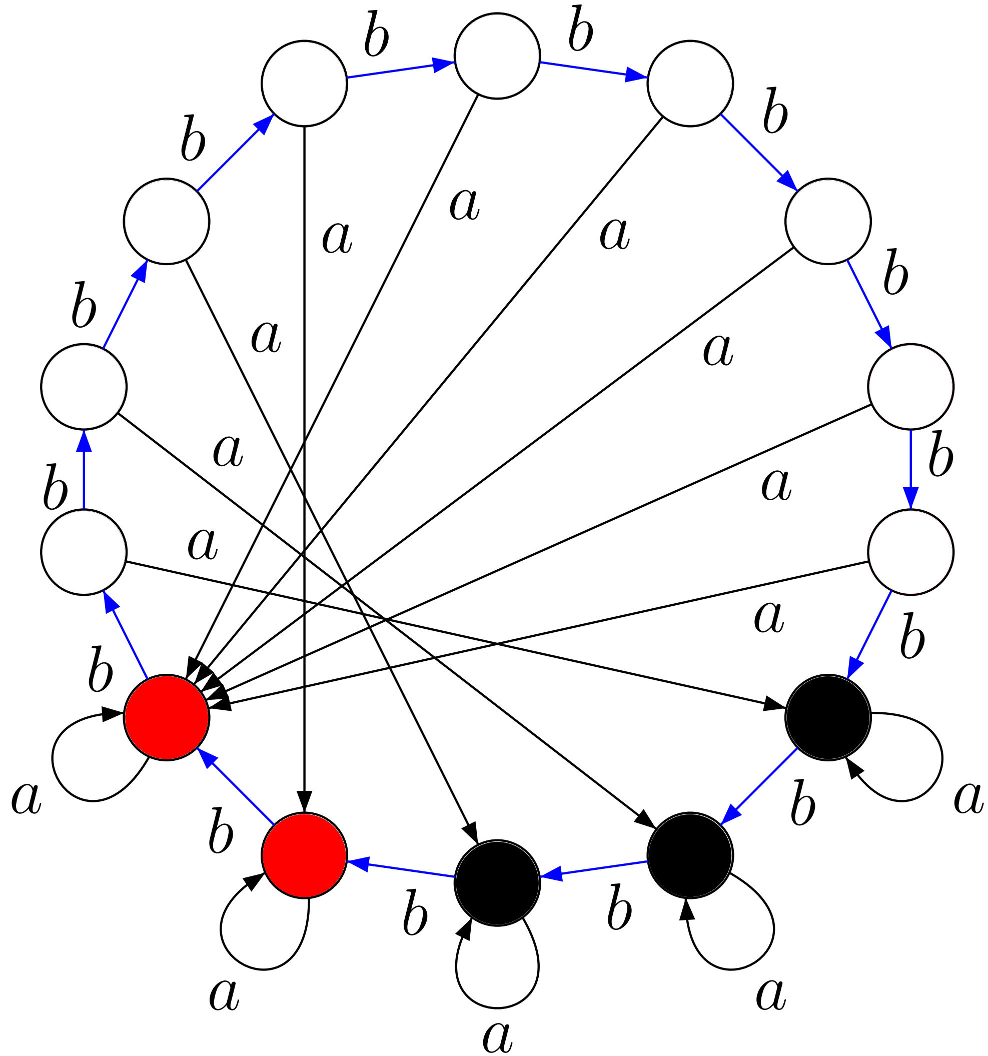


“Honest” example: “greedy” reset word (k=2)



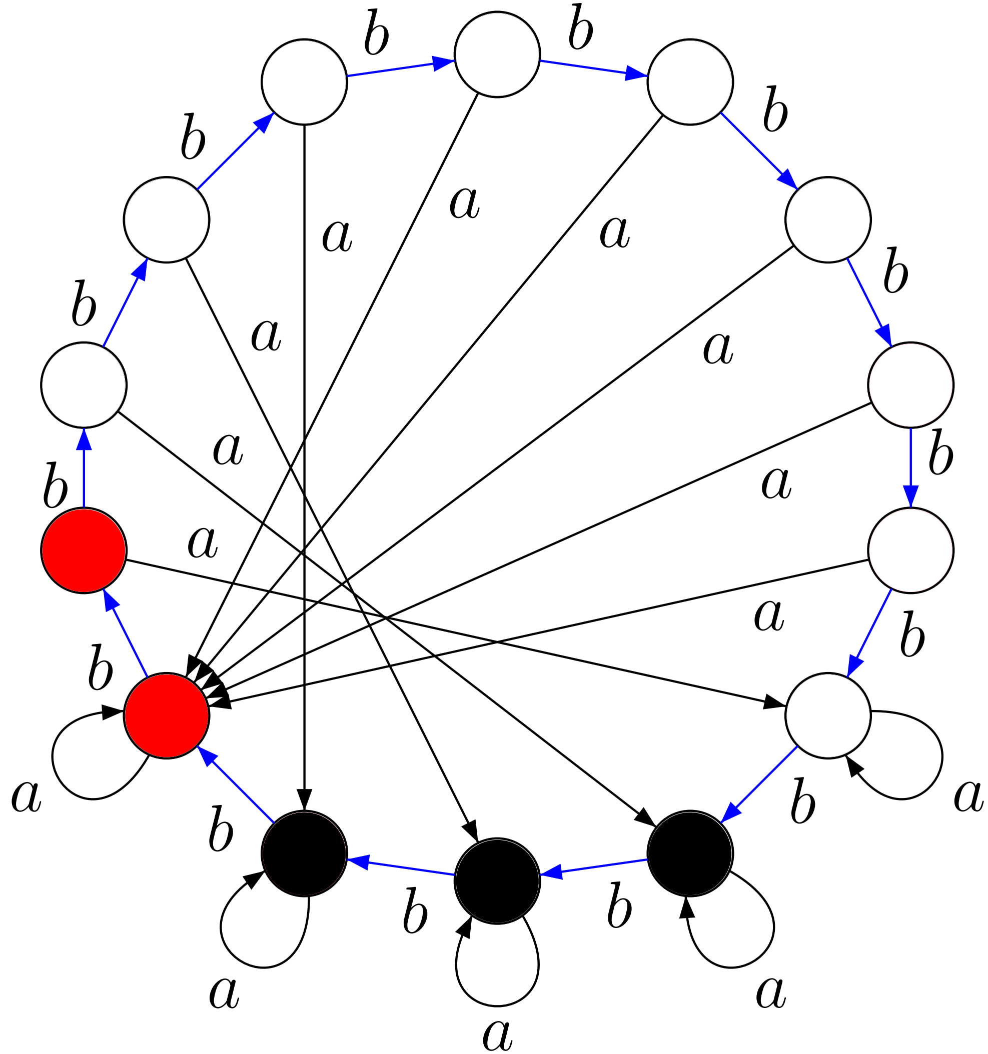
a

“Honest” example: “greedy” reset word (k=2)



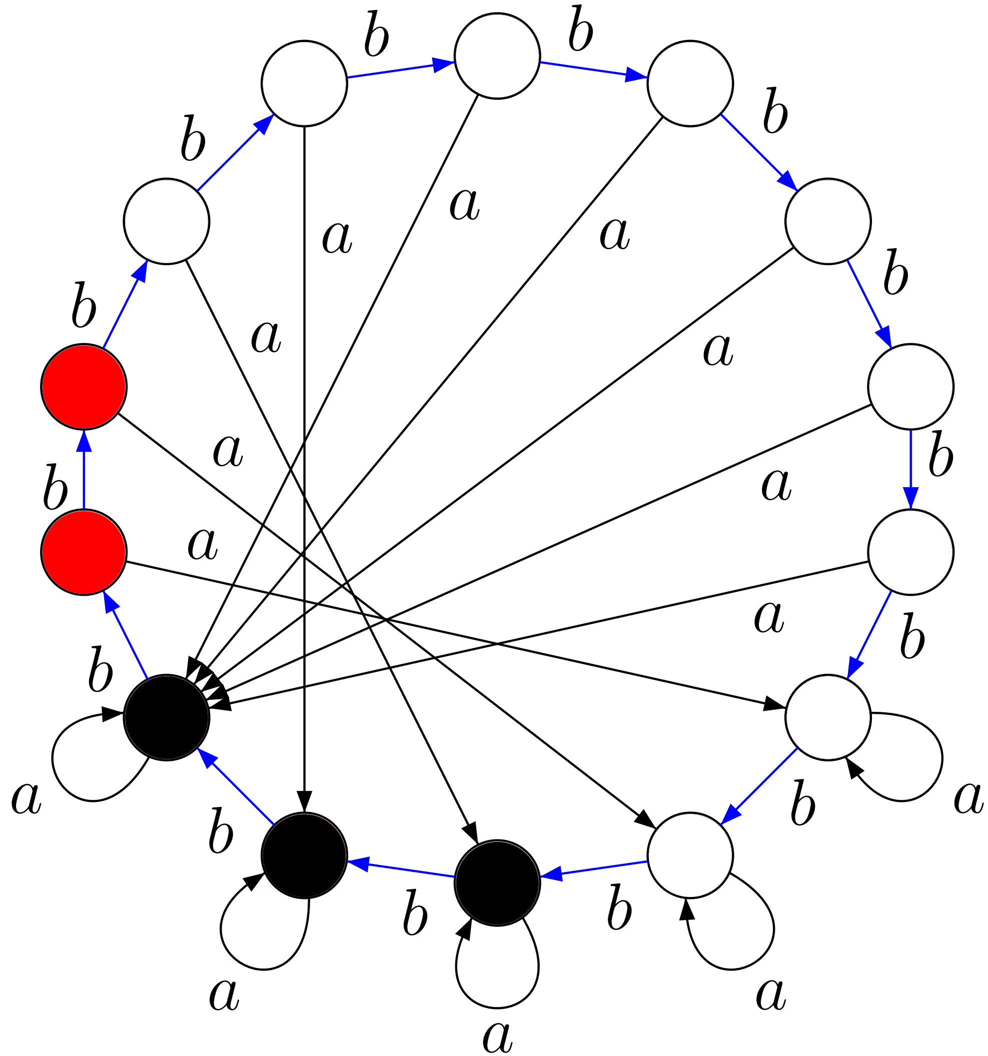
$abbbbbba$

“Honest” example: “greedy” reset word (k=2)



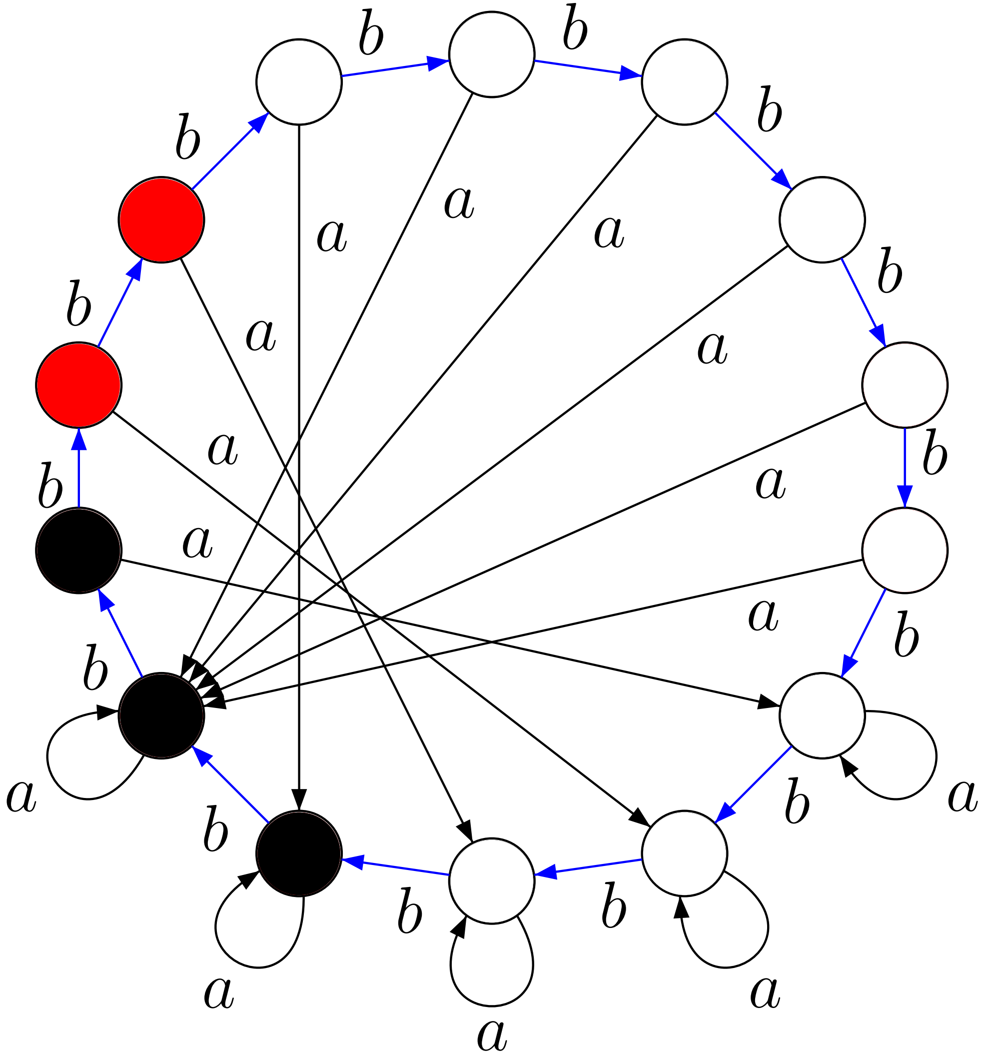
$a\textcolor{red}{b}bbbbba$

“Honest” example: “greedy” reset word (k=2)



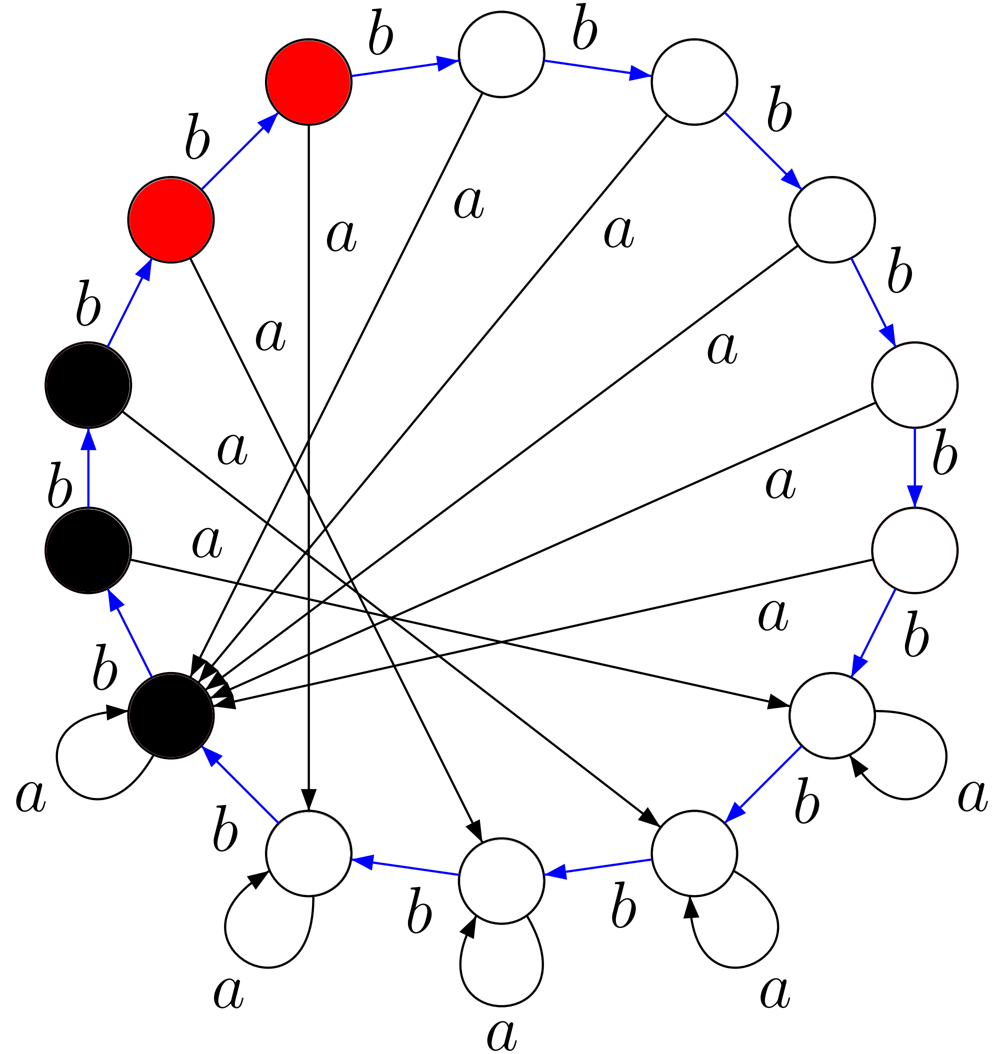
$abbbbbba$

"Honest" example: "greedy" reset word (k=2)



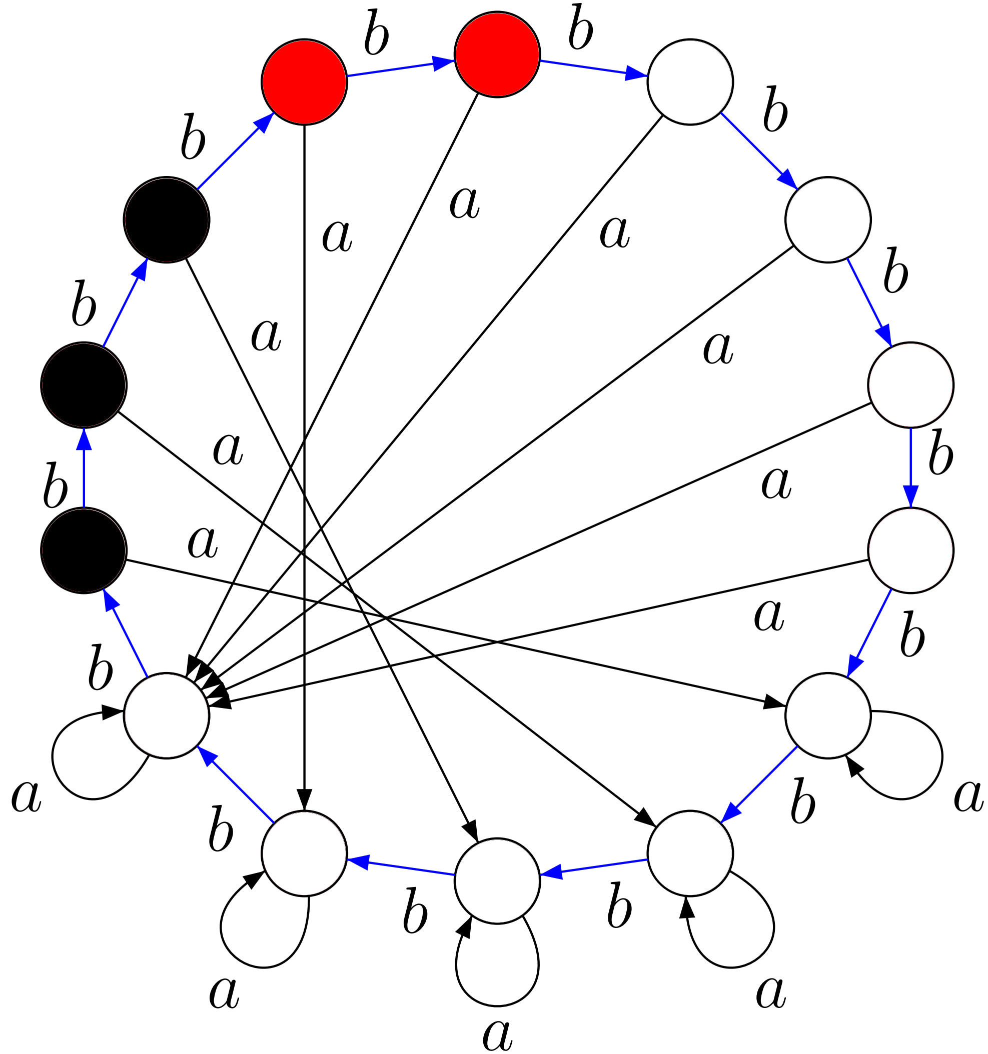
$abb\textcolor{red}{b}bbba$

“Honest” example: “greedy” reset word (k=2)



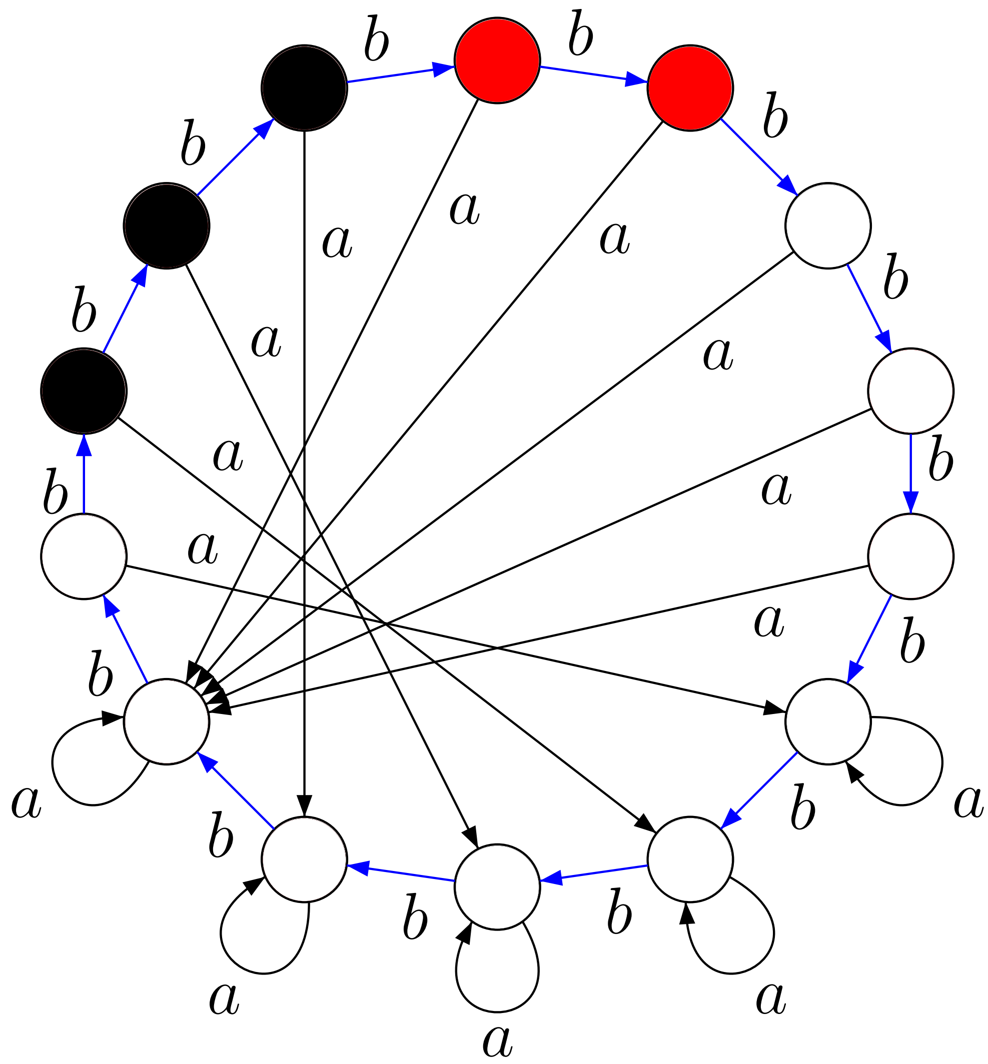
$abbbbbba$

“Honest” example: “greedy” reset word (k=2)



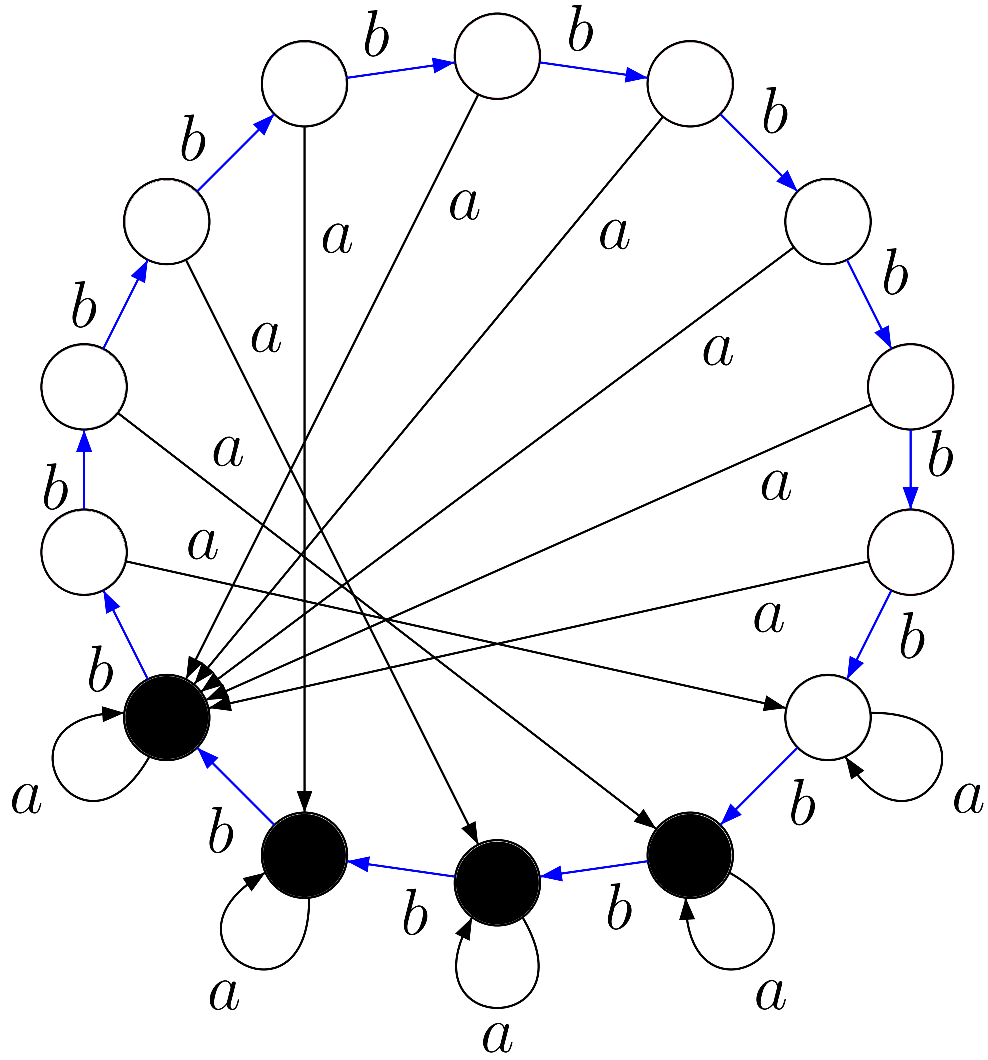
$abbbbbba$

"Honest" example: "greedy" reset word (k=2)



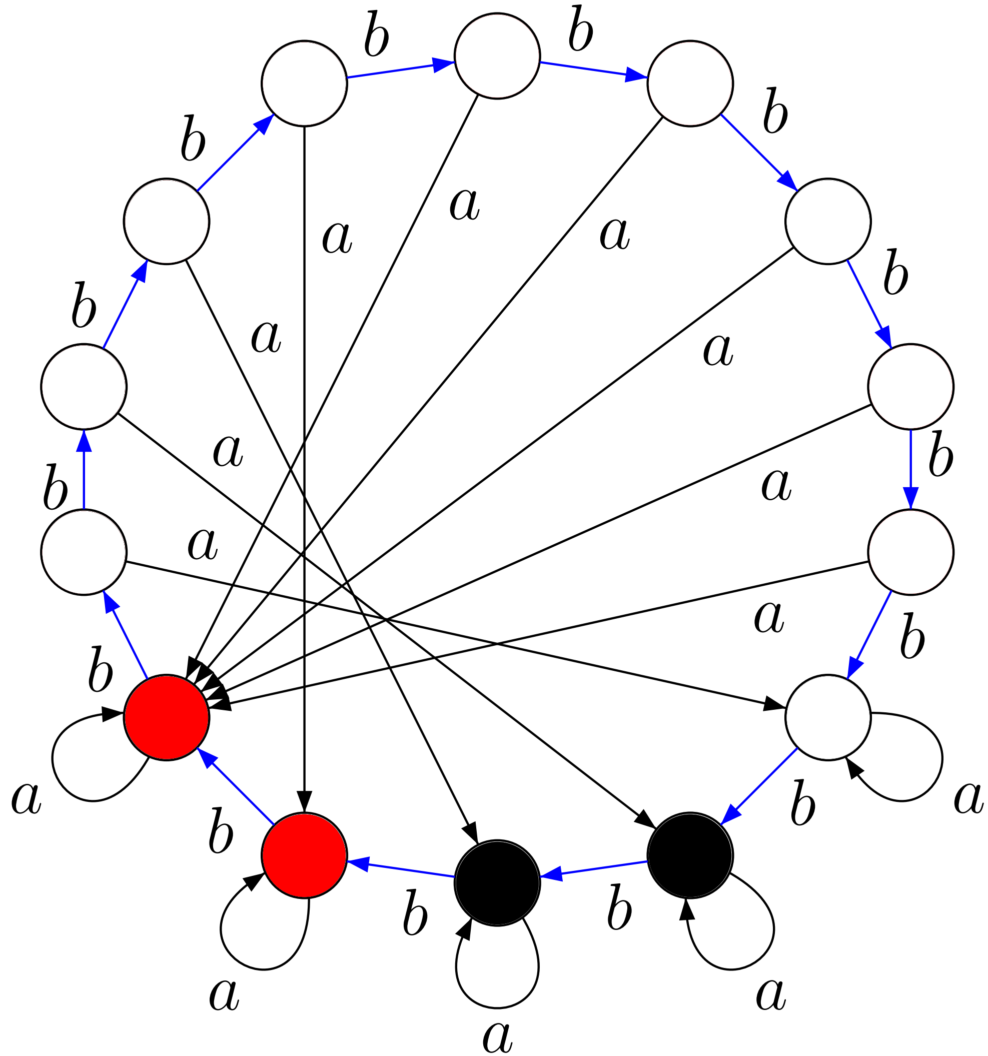
$abbbbbba$

“Honest” example: “greedy” reset word (k=2)



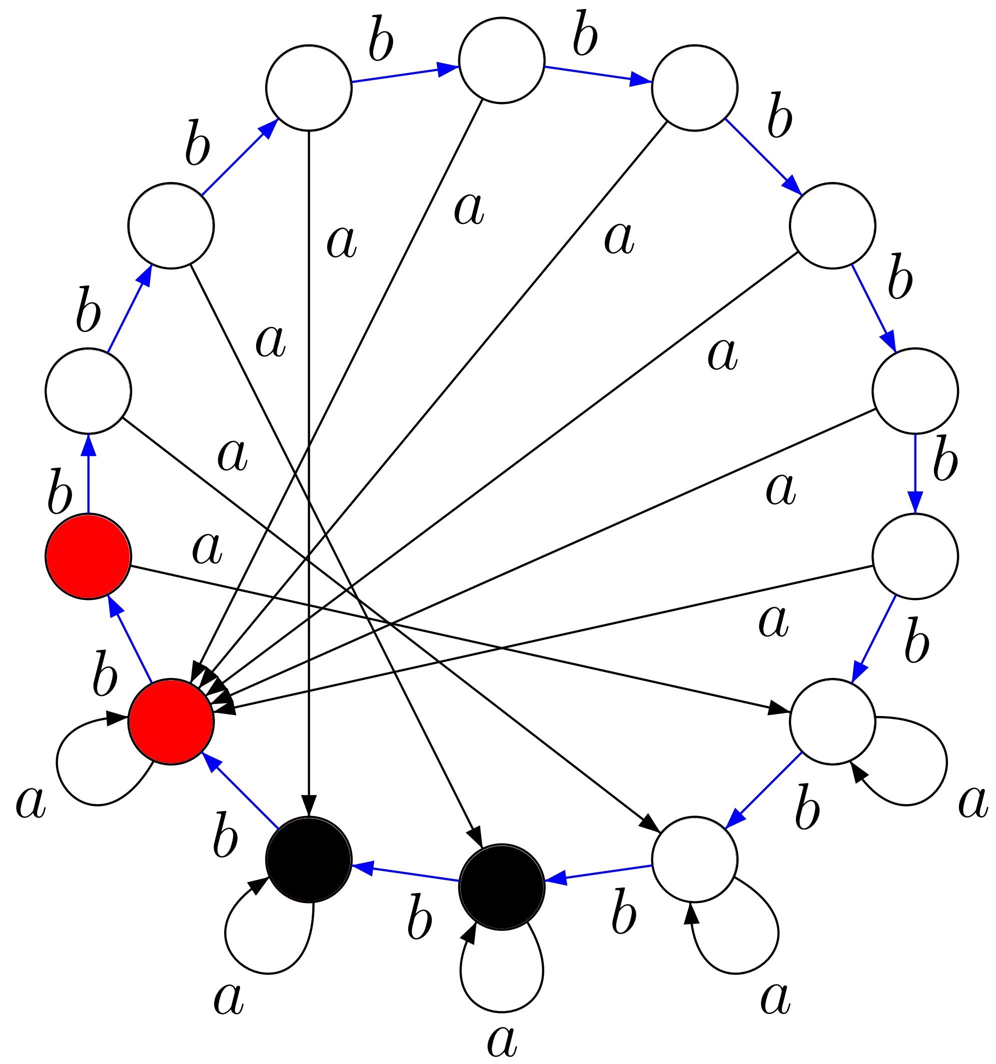
$abbbbbba$

“Honest” example: “greedy” reset word (k=2)



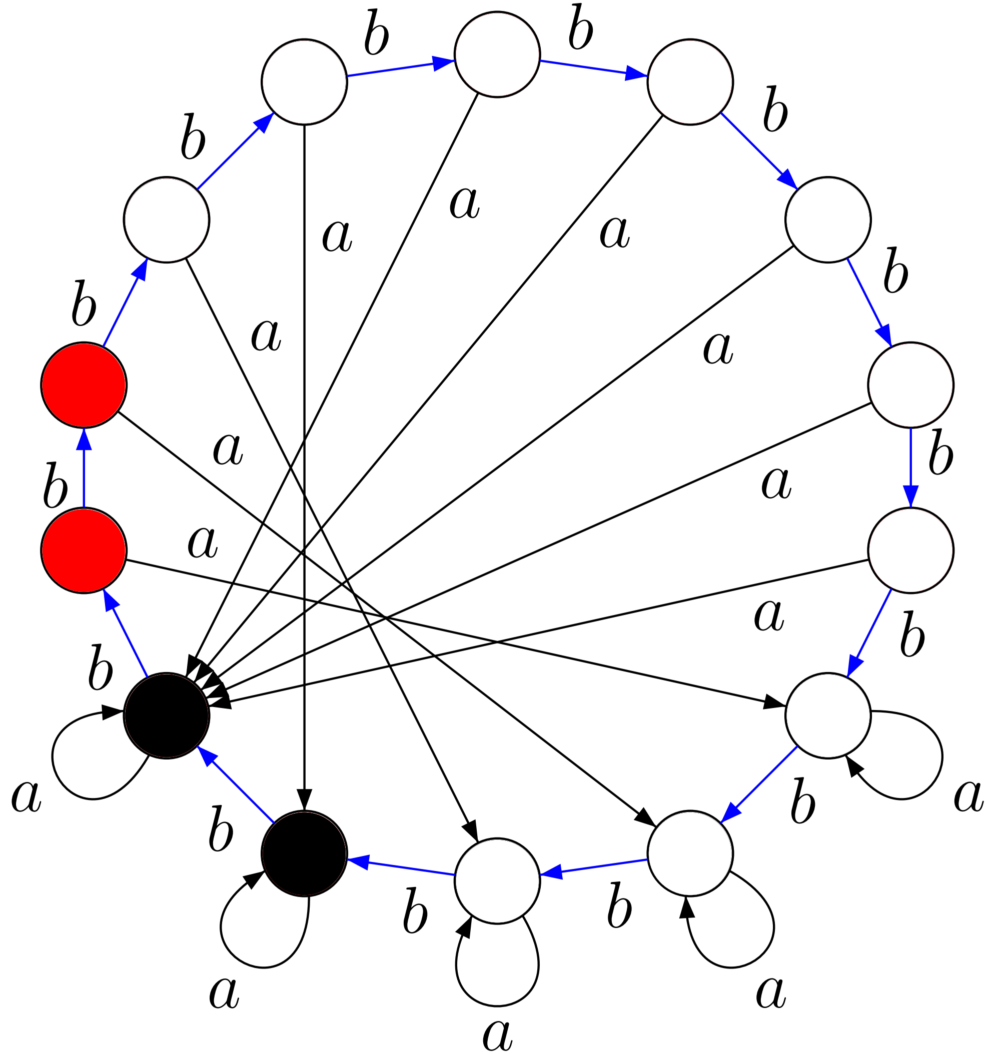
$abbbbbb{\color{red}a}bbbbbba$

“Honest” example: “greedy” reset word (k=2)



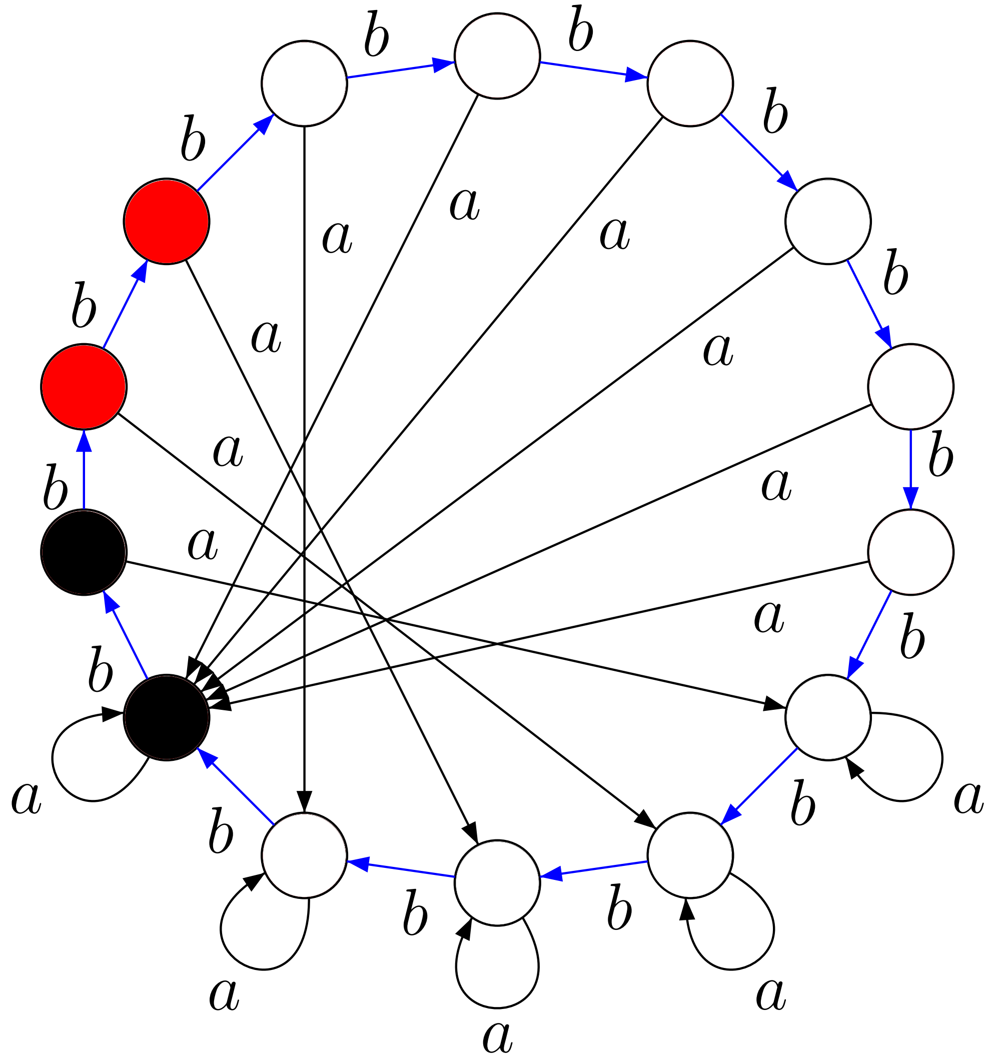
$abbbbbbabbbbbba$

"Honest" example: "greedy" reset word (k=2)



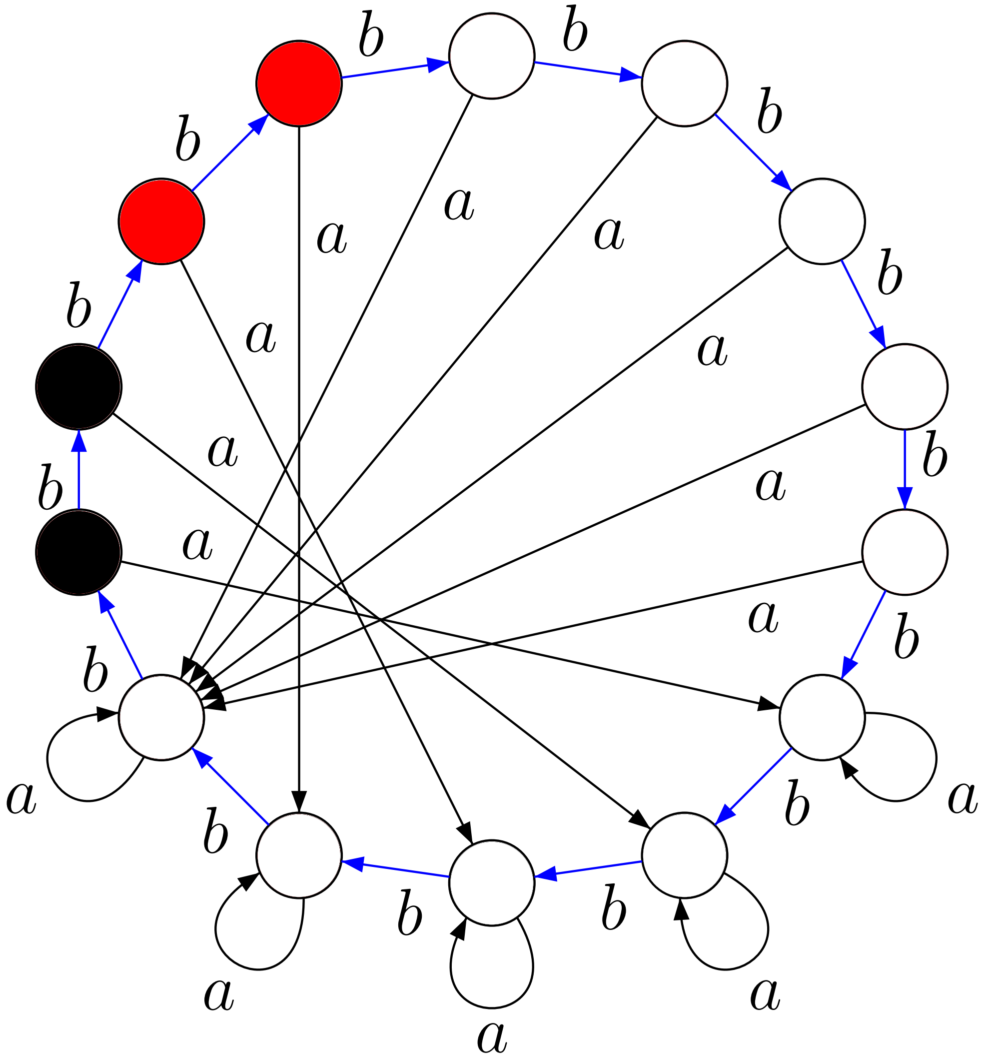
$abbbbbbabbbbbba$

“Honest” example: “greedy” reset word (k=2)



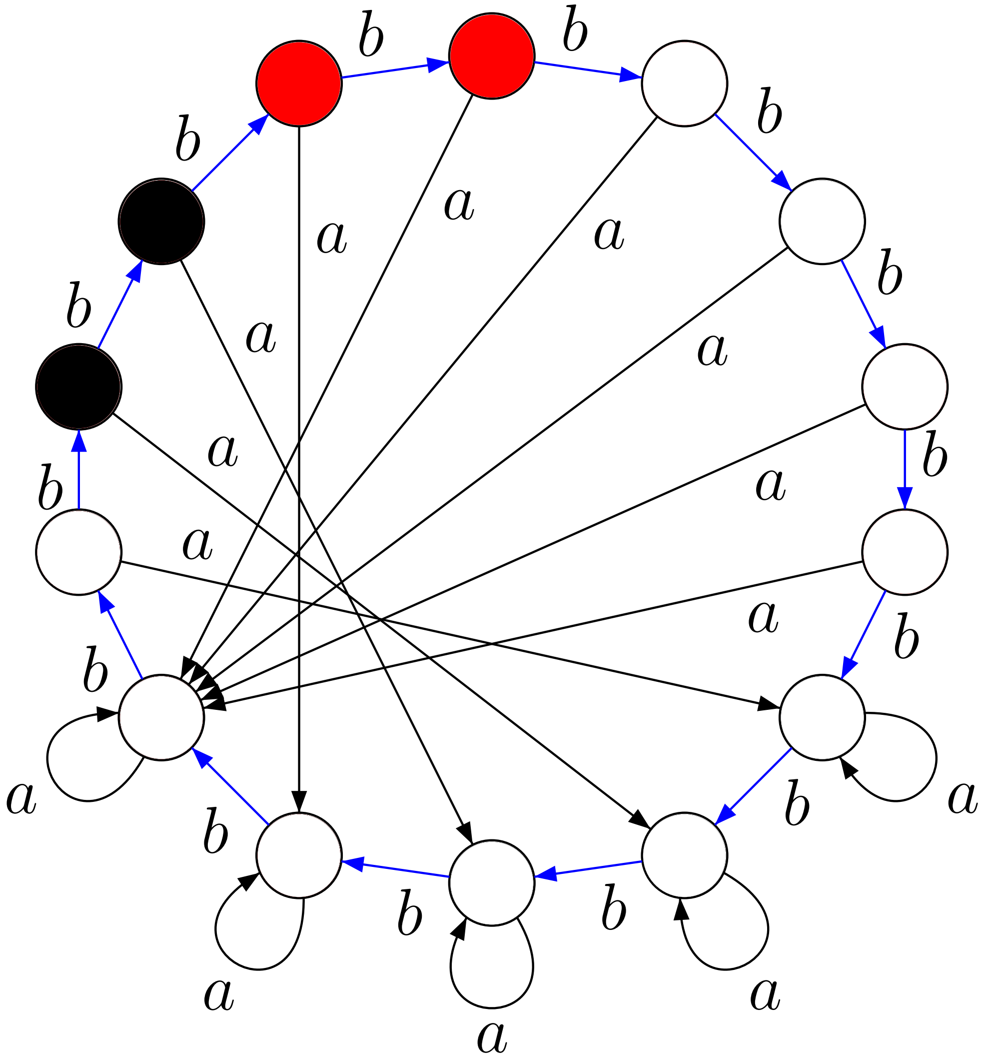
$abbbbbbabbbbbba$

"Honest" example: "greedy" reset word (k=2)



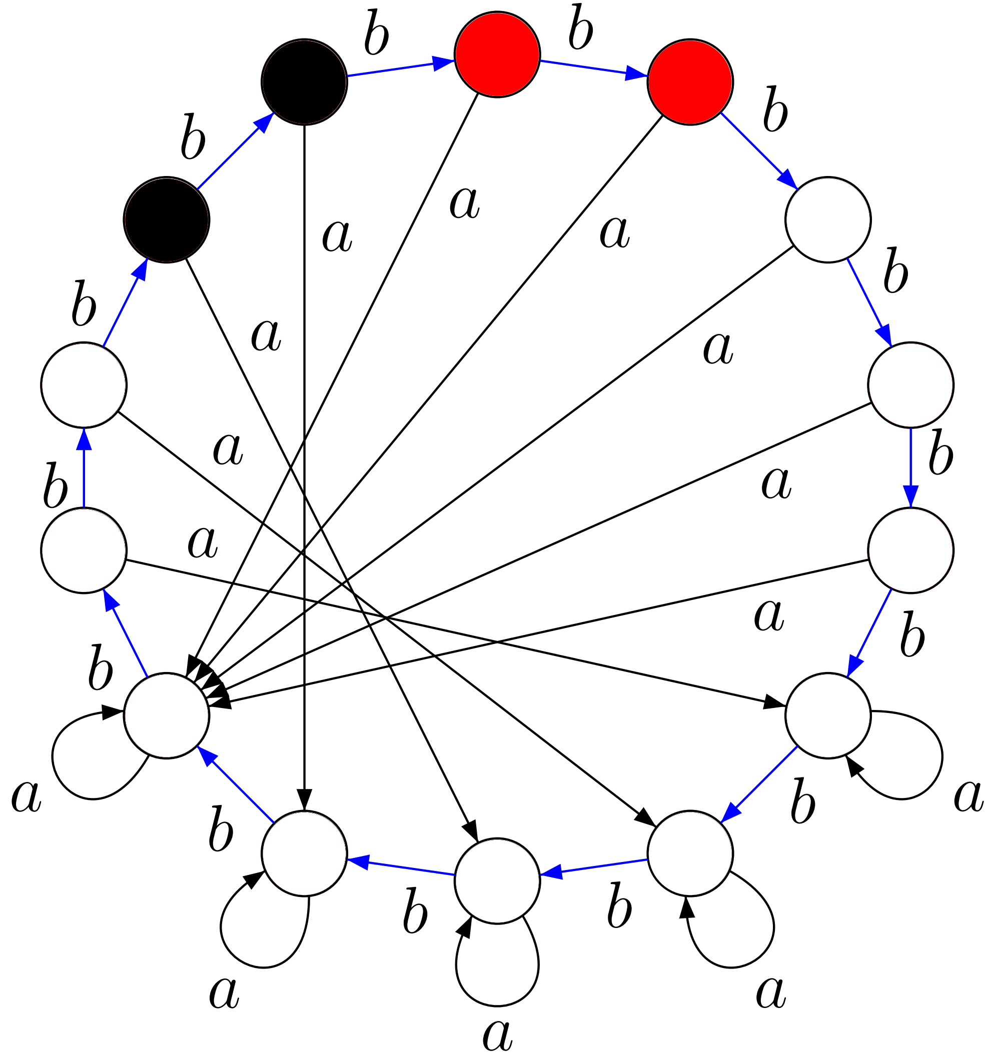
$abbbbbbabbb{\color{red}b}bba$

"Honest" example: "greedy" reset word (k=2)



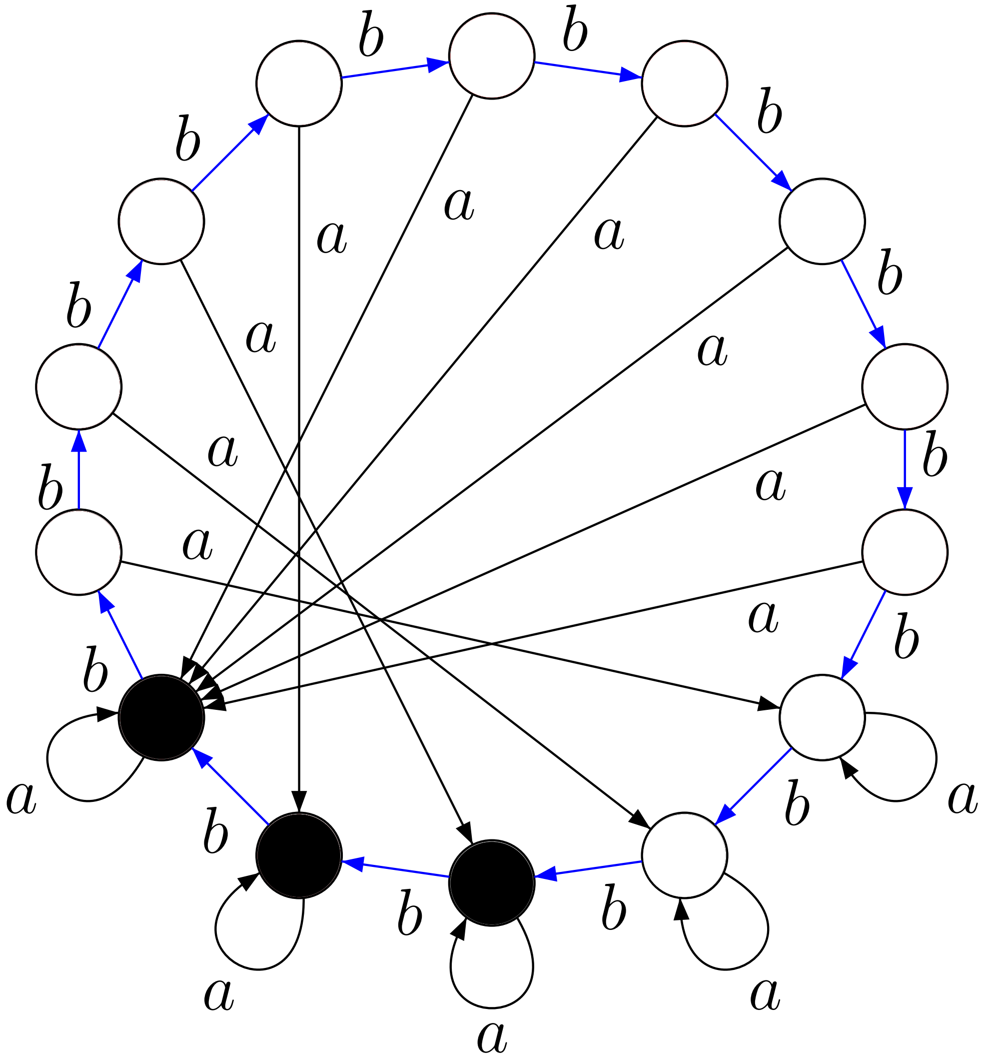
$abbbbbbabbbbbba$

“Honest” example: “greedy” reset word (k=2)



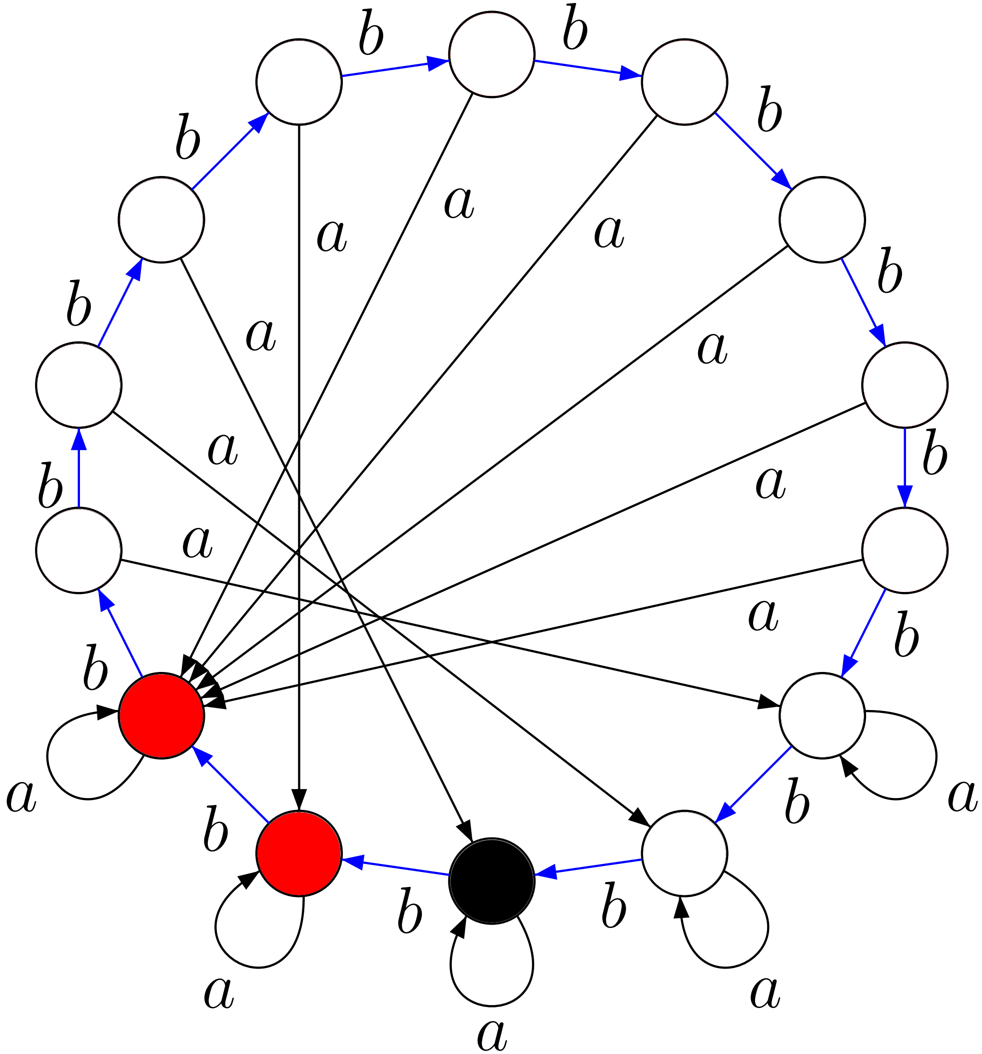
$abbbbbbabbbbbba$

“Honest” example: “greedy” reset word (k=2)



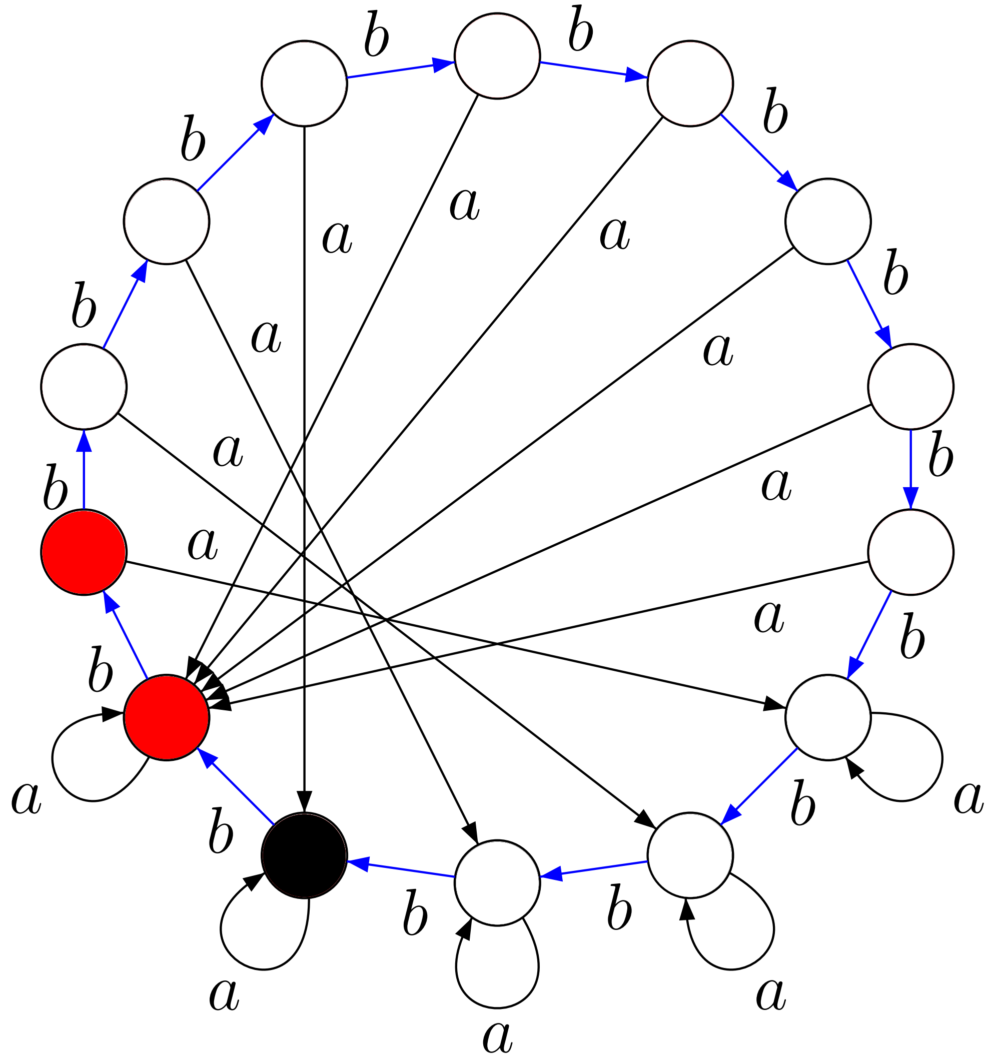
$abbbbbbabbbbbb\textcolor{red}{a}$

“Honest” example: “greedy” reset word (k=2)



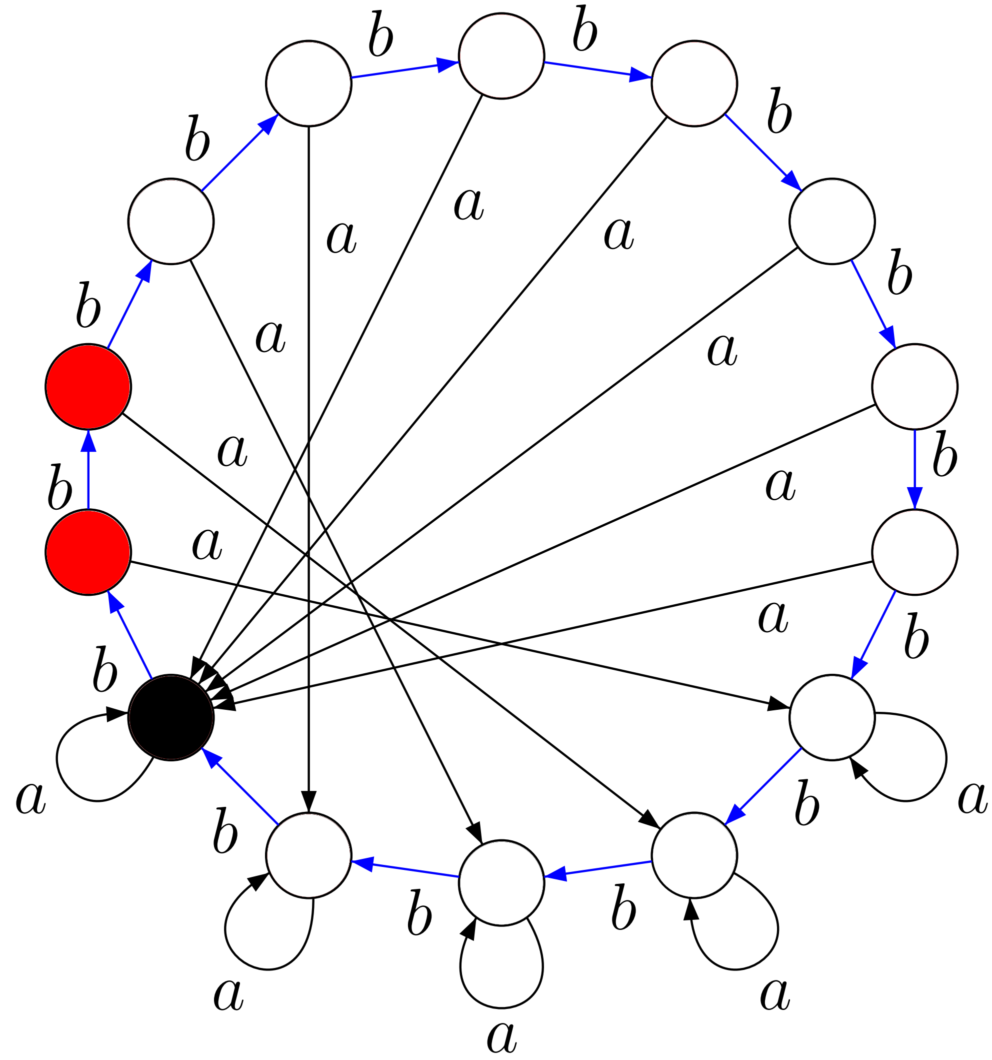
$abbbbbbabbbbbbabbbbbba$

“Honest” example: “greedy” reset word (k=2)



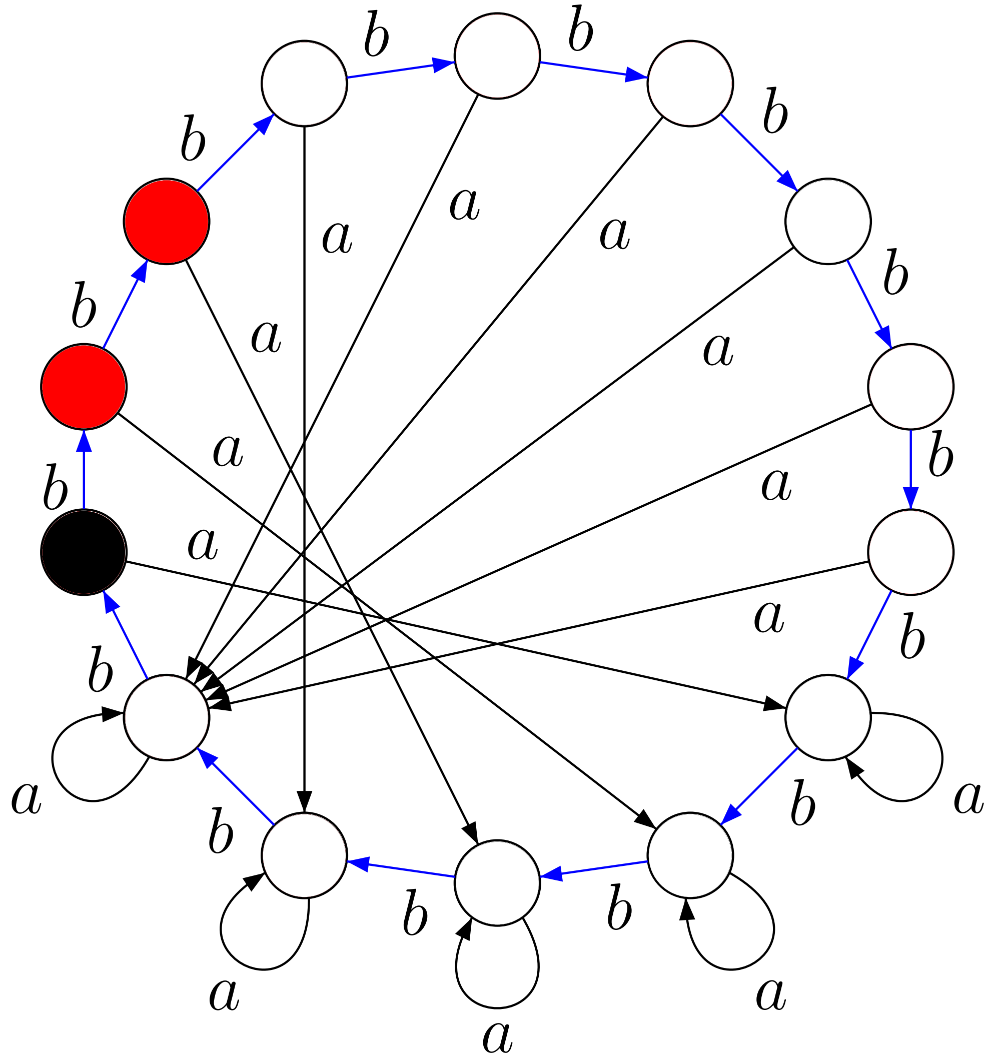
$abbbbbbabbbbbbabbbbbba$

"Honest" example: "greedy" reset word (k=2)



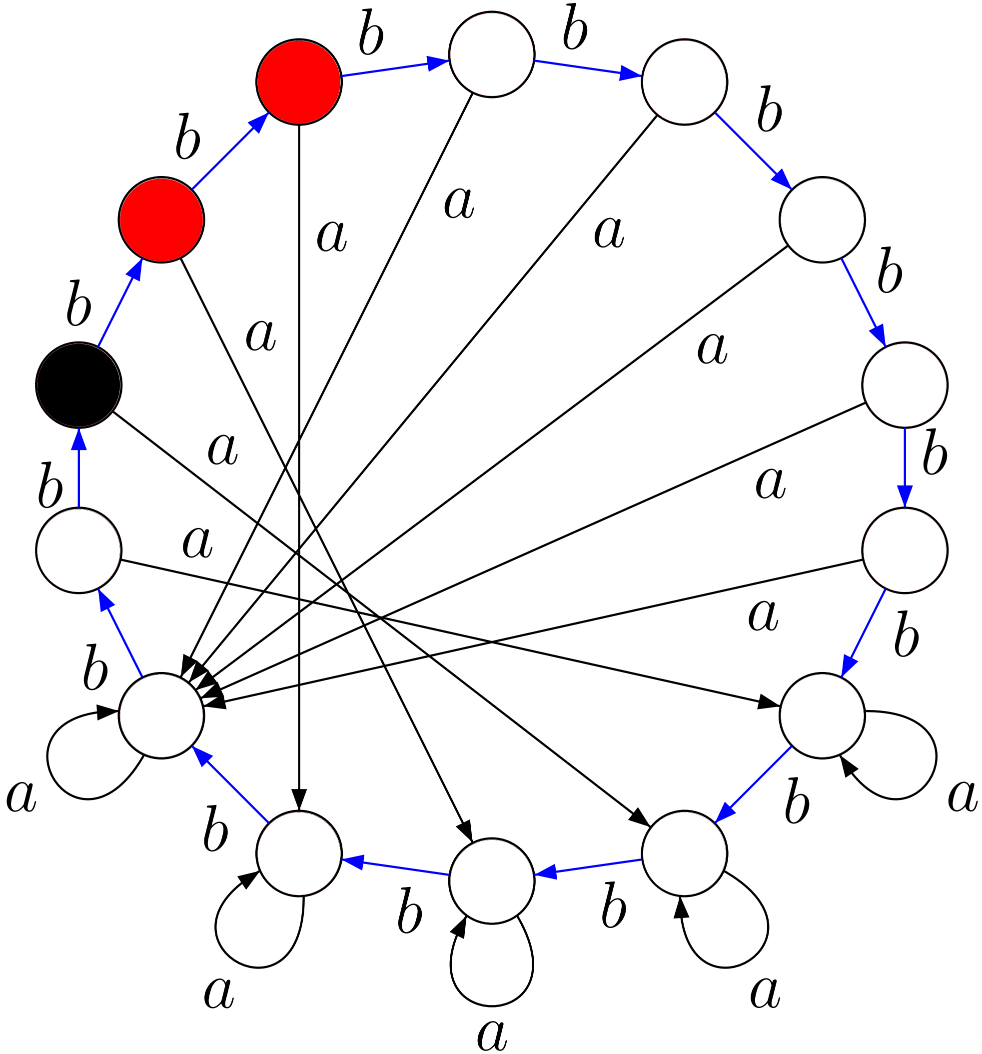
$abbbbbbabbbbbbabbbbbba$

“Honest” example: “greedy” reset word (k=2)



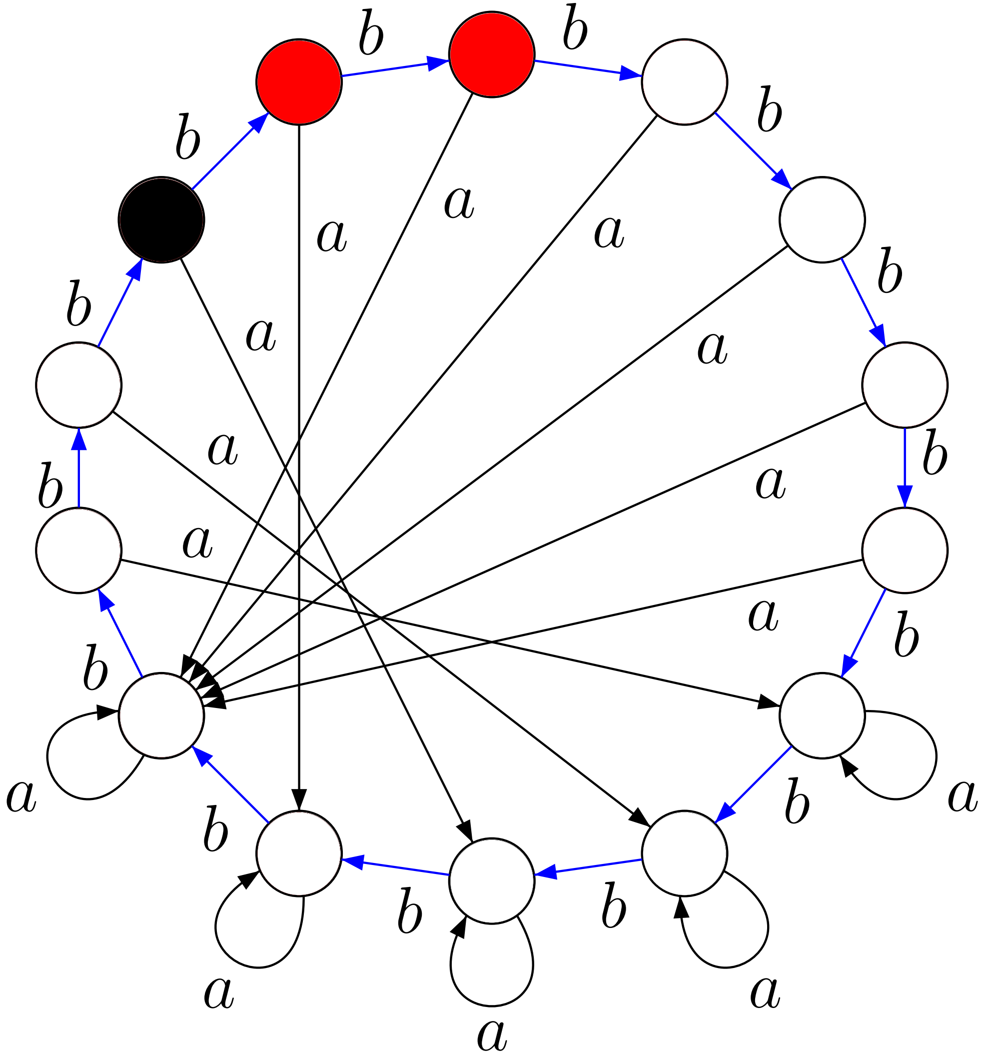
$abbbbbbabbbbbbabb{\color{red}b}bbba$

“Honest” example: “greedy” reset word (k=2)



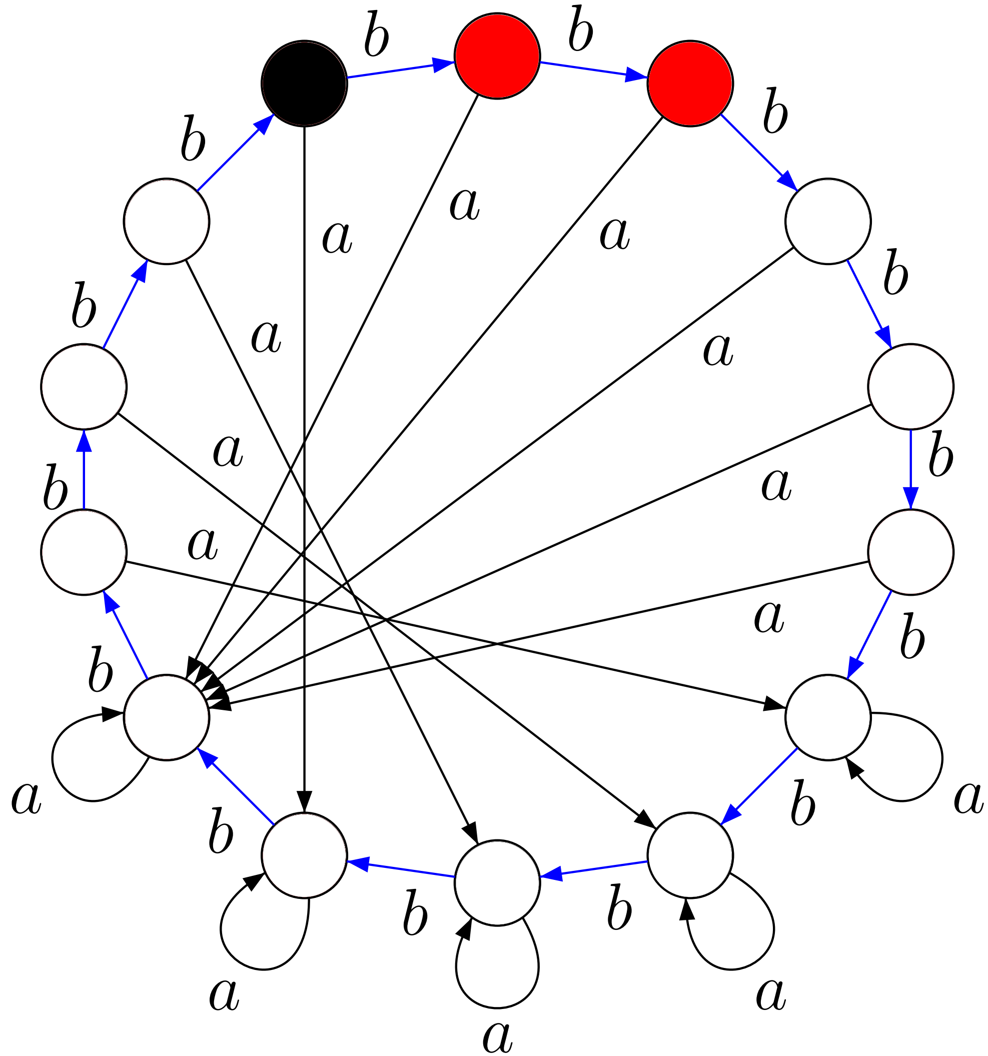
$abbbbbbabbbbbbabbbbbba$

"Honest" example: "greedy" reset word (k=2)



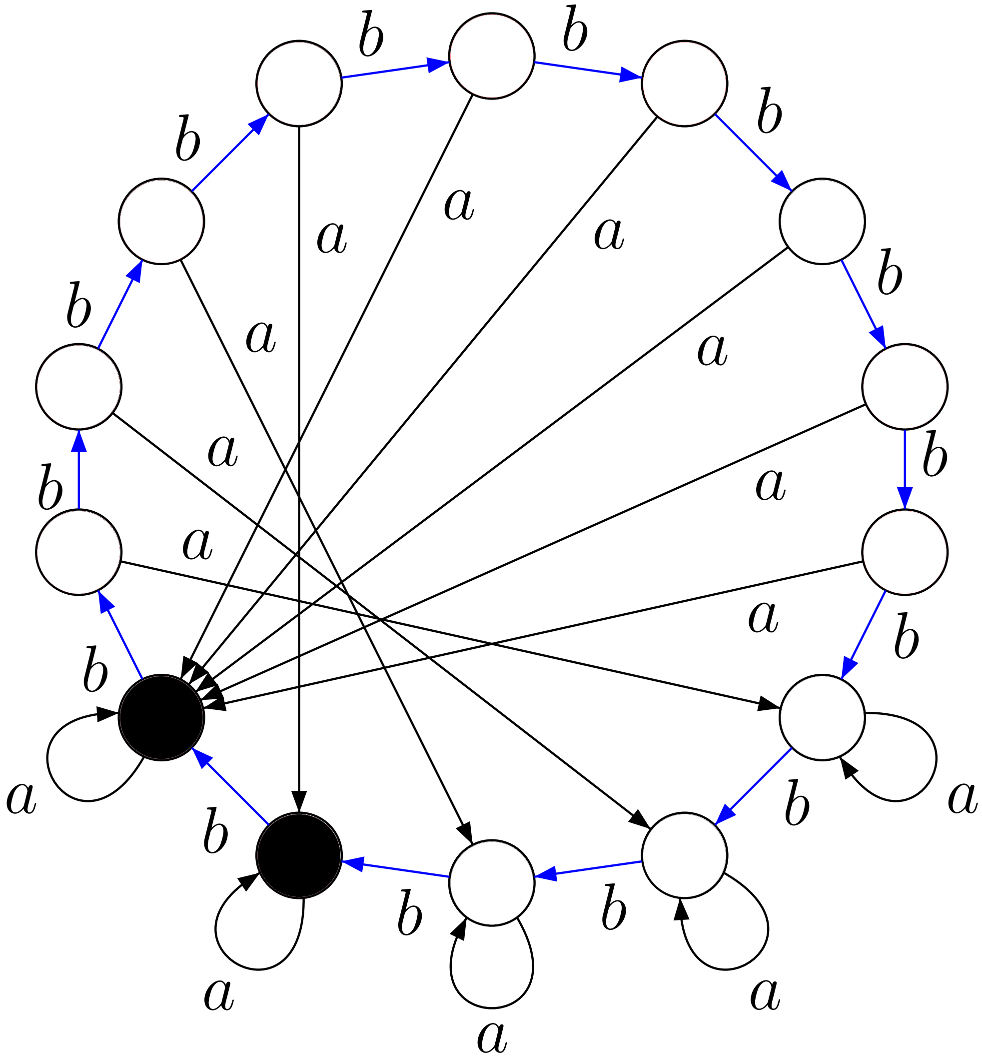
$abbbbbbabbbbbbabbbbbba$

“Honest” example: “greedy” reset word (k=2)



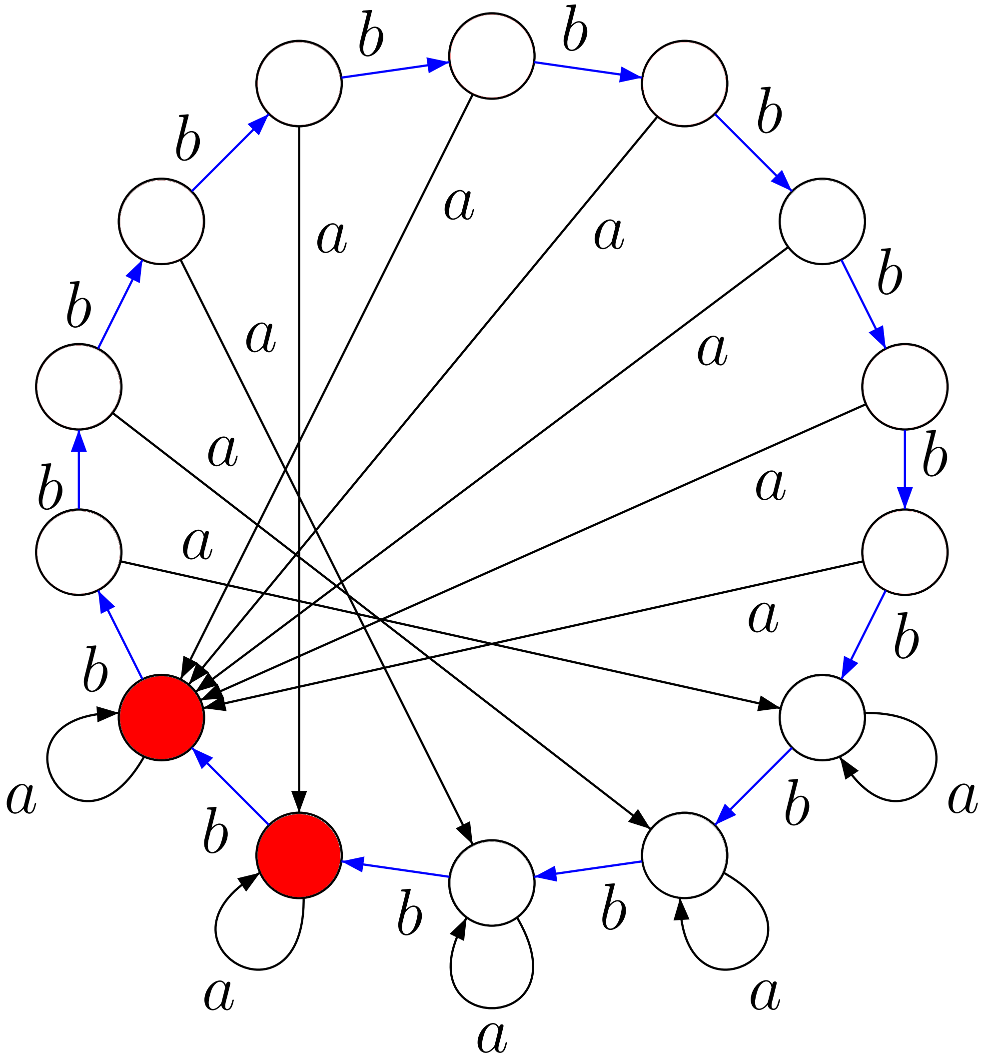
$abbbbbbabbbbbbabbbbbba$

“Honest” example: “greedy” reset word (k=2)



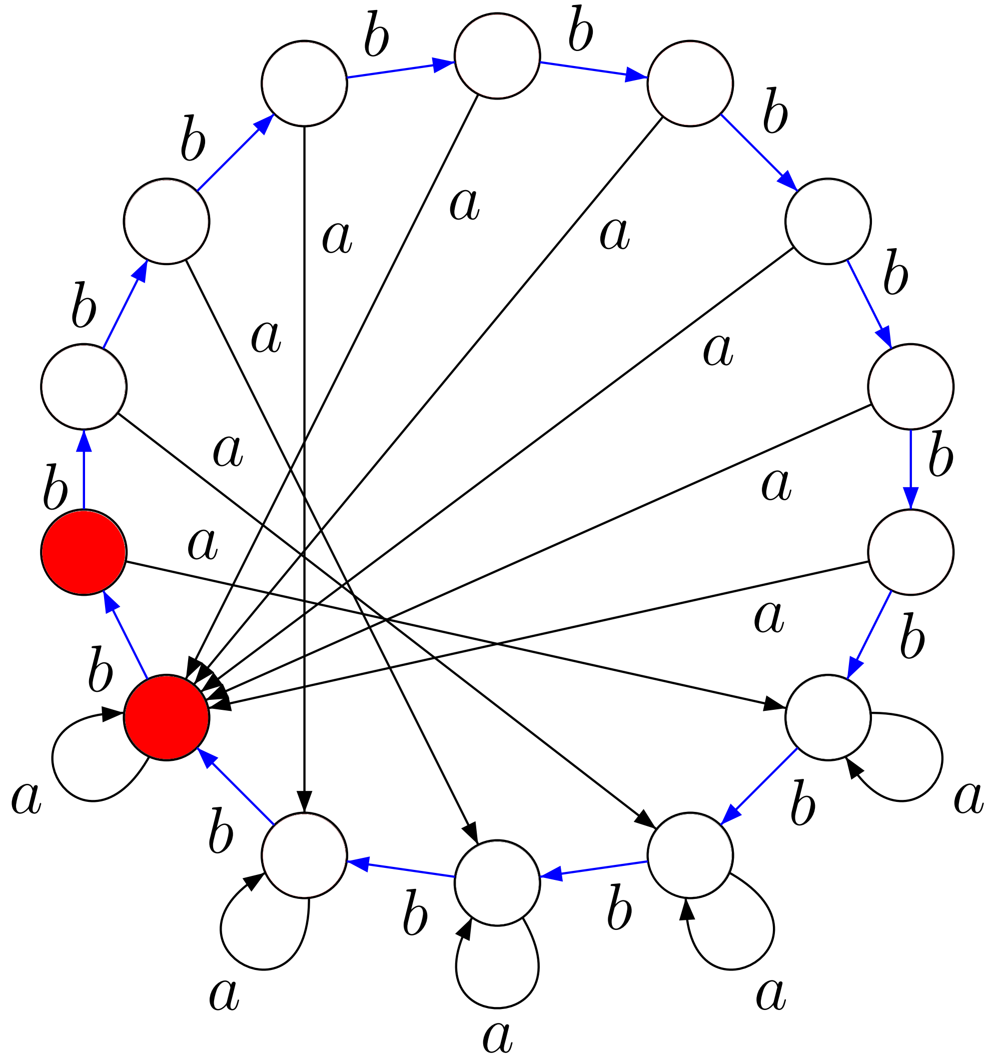
$abbbbbbabbbbbbabbbbbb\textcolor{red}{a}$

“Honest” example: “greedy” reset word (k=2)



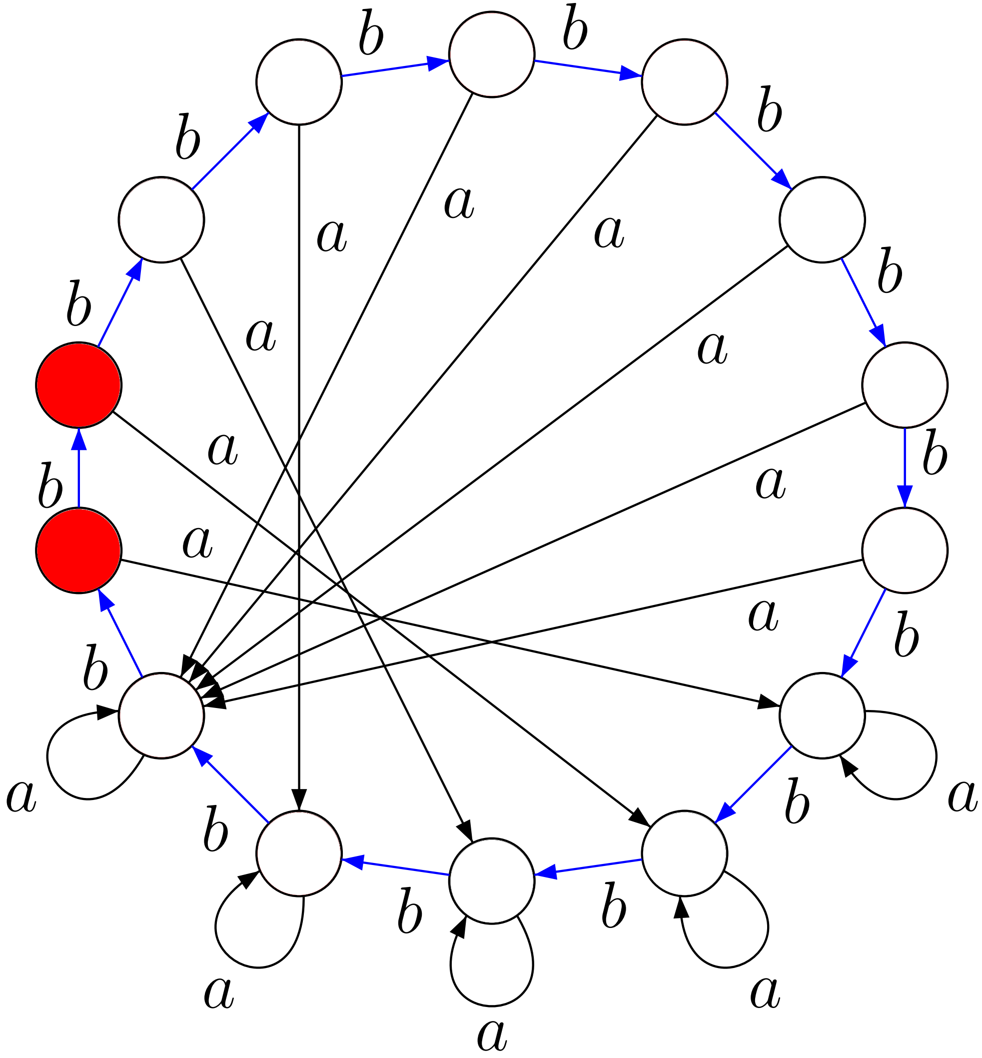
$abbbbbbabbbbbbabbbbbb{\color{red}a}bbbbbba$

“Honest” example: “greedy” reset word (k=2)



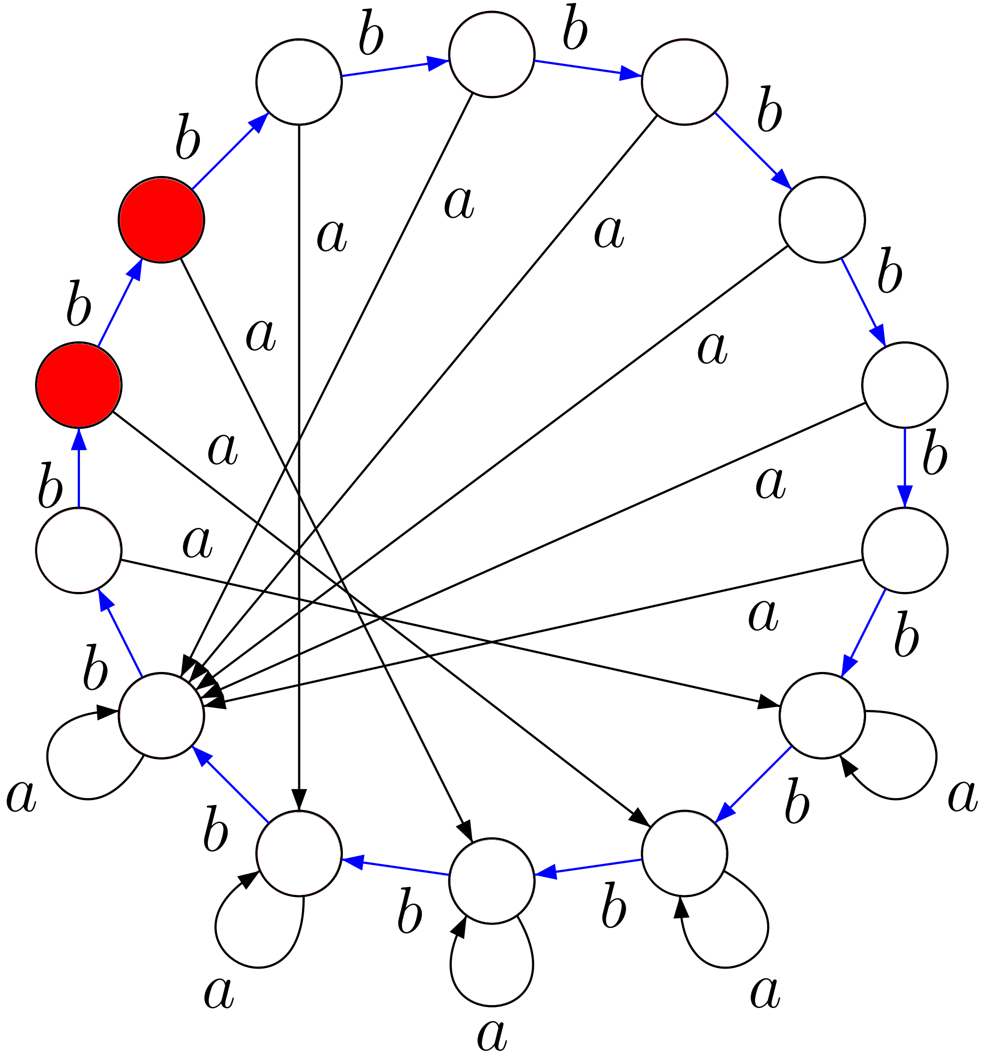
$abbbbbbabbbbbbabbbbbba\textcolor{red}{b}bbbbba$

"Honest" example: "greedy" reset word (k=2)



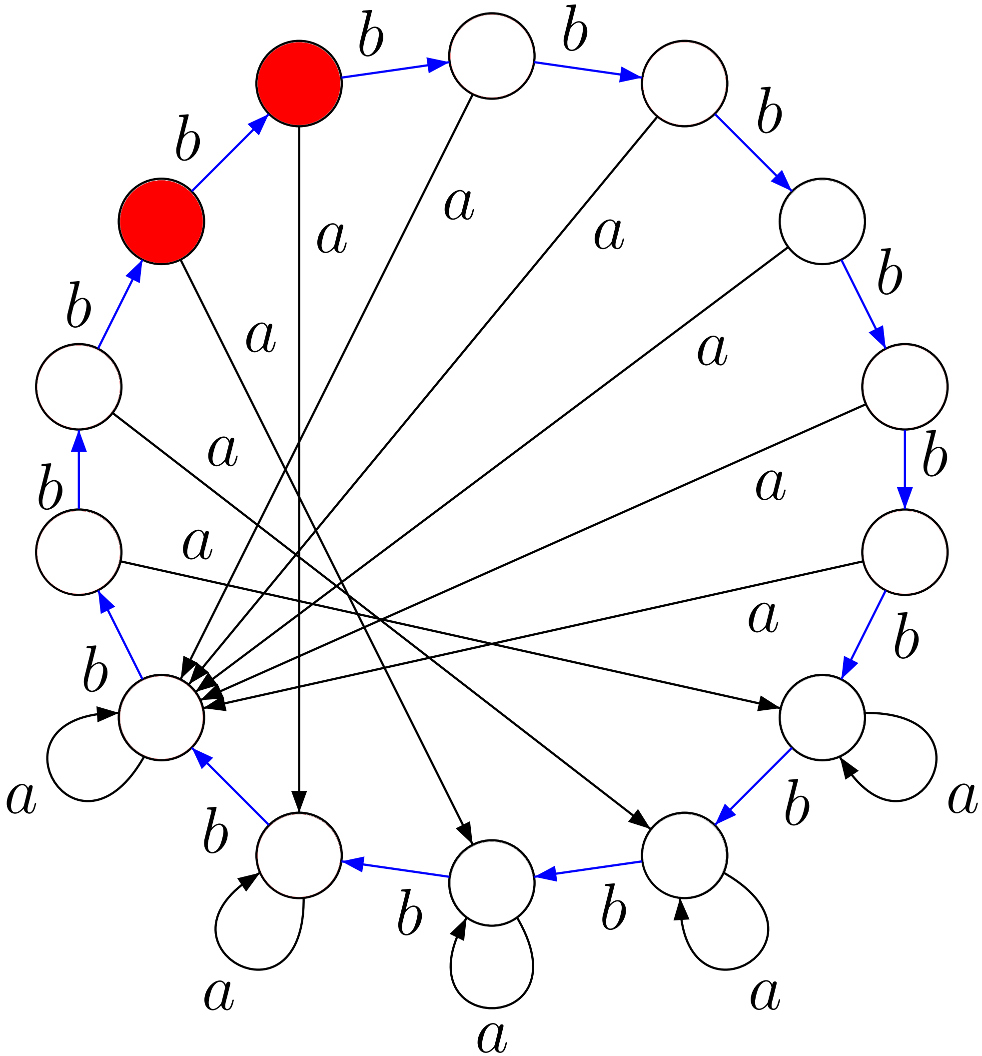
$abbbbbbabbbbbbabbbbbbab{\color{red}b}bbbba$

"Honest" example: "greedy" reset word (k=2)



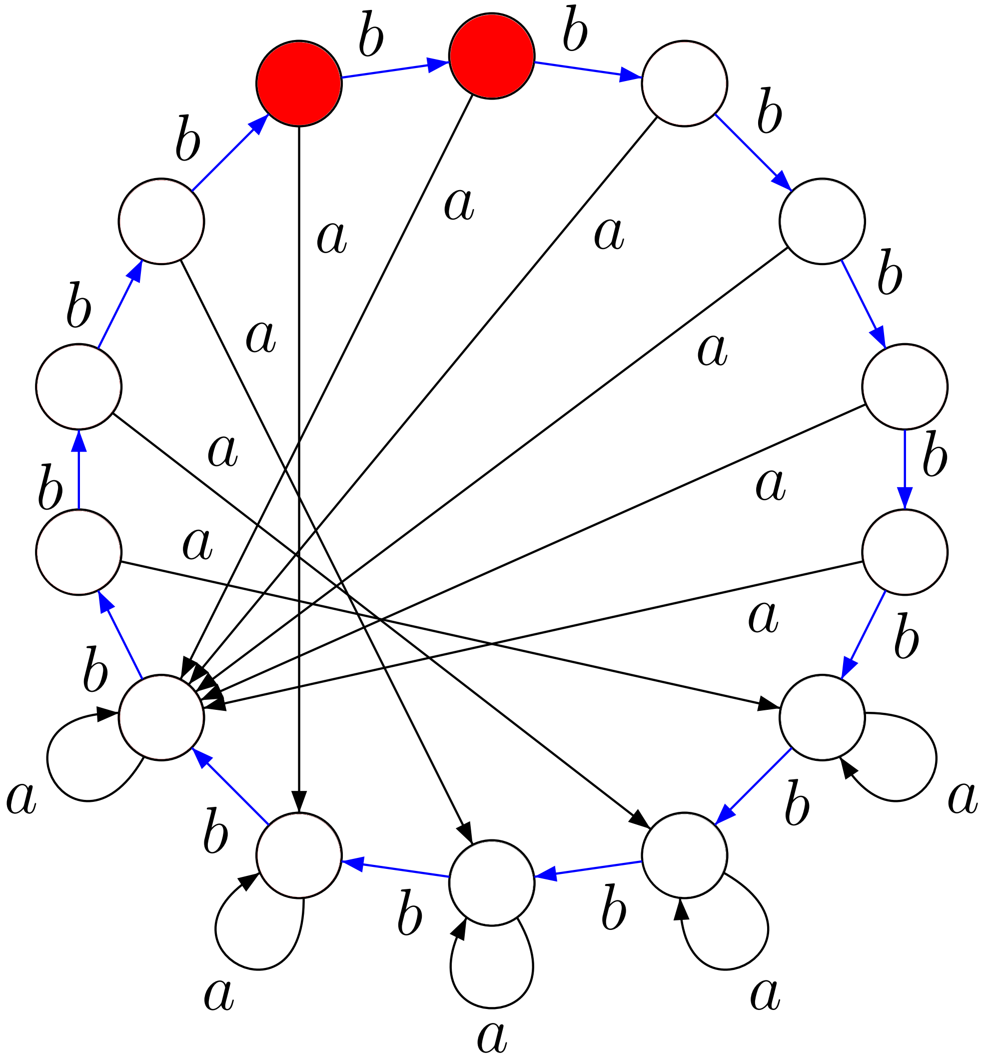
$abbbbbbabbbbbbabbbbbbabbbbbba$

“Honest” example: “greedy” reset word (k=2)



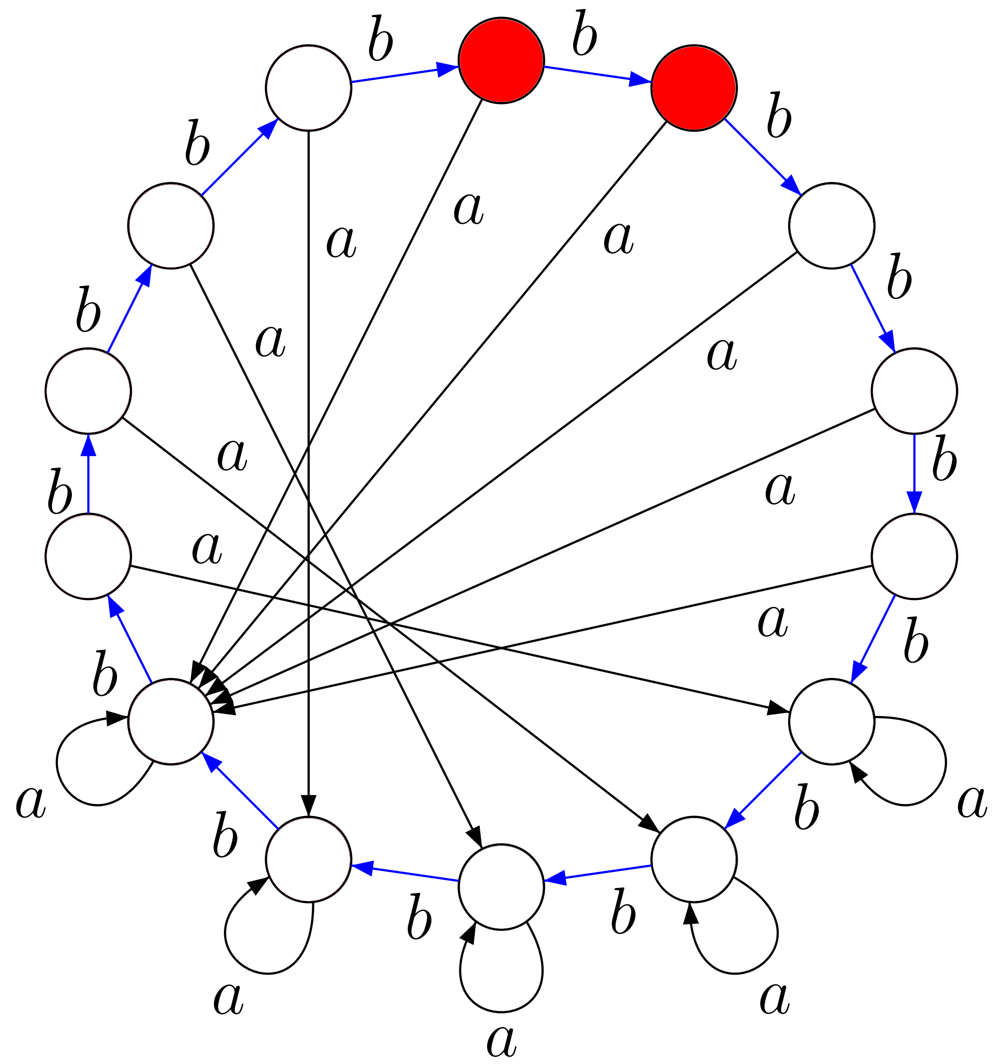
$abbbbbbabbbbbbabbbbbbabbb\textcolor{red}{b}bba$

“Honest” example: “greedy” reset word (k=2)



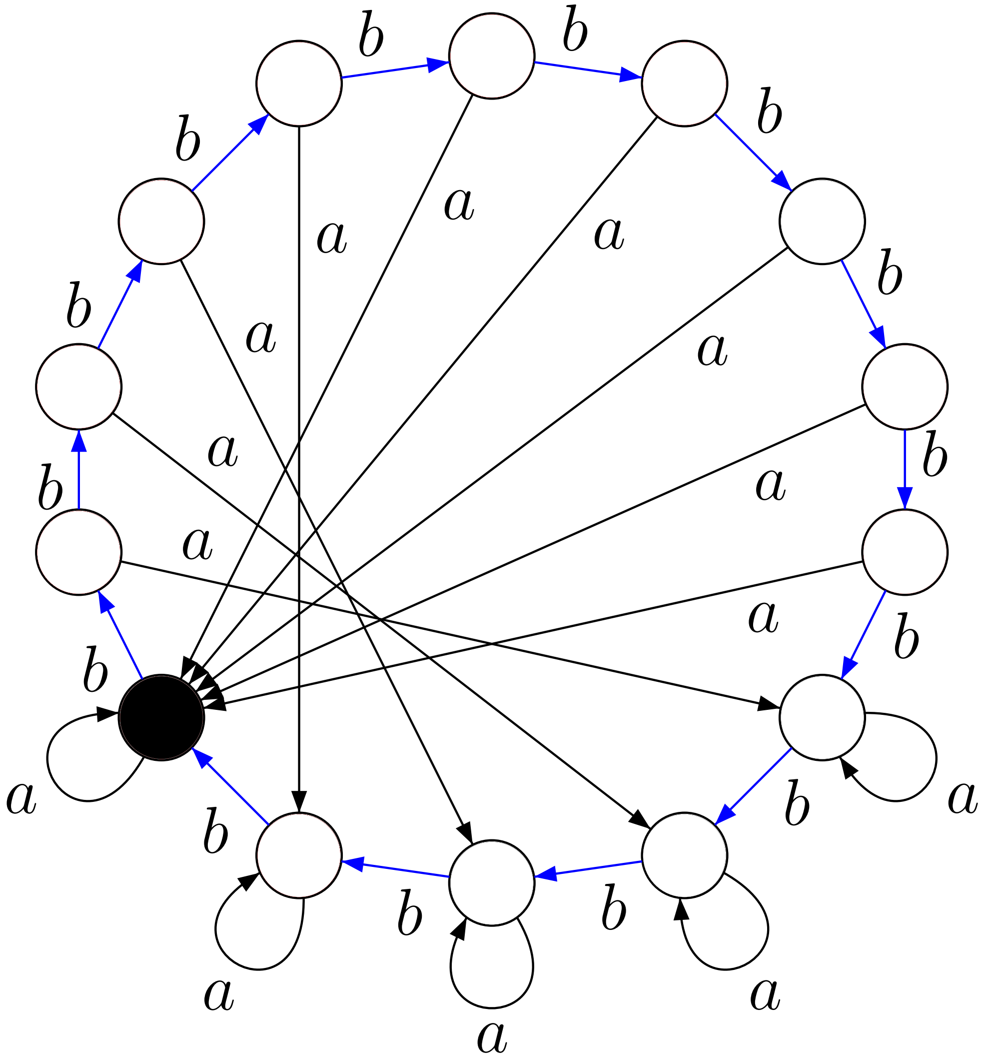
$abbbbbbabbbbbbabbbbbbabbbb{\color{red}b}ba$

"Honest" example: "greedy" reset word (k=2)



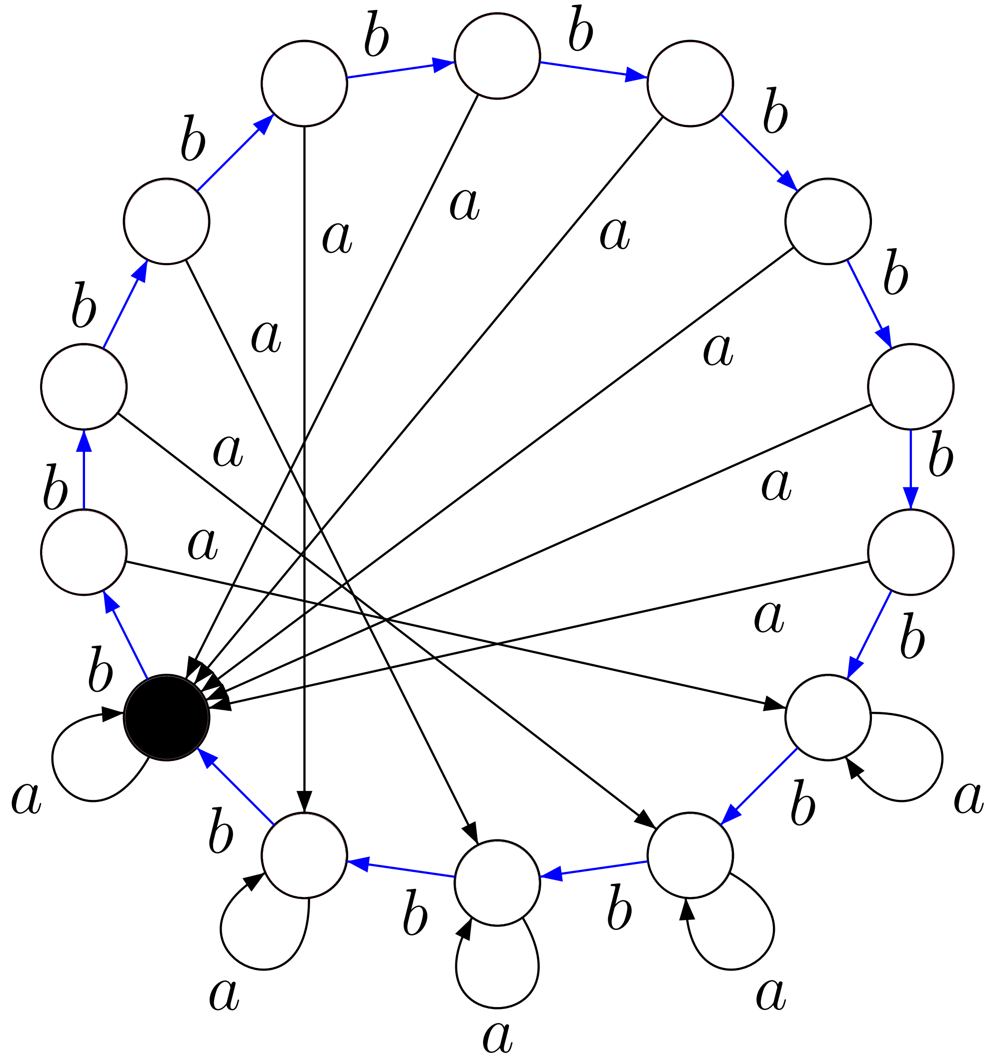
$abbbbbbabbbbbbabbbbbbabbbbb\textcolor{red}{b}a$

“Honest” example: “greedy” reset word (k=2)



$abbbbbbabbbbbbabbbbbbabbbbbb{\color{red}a}$

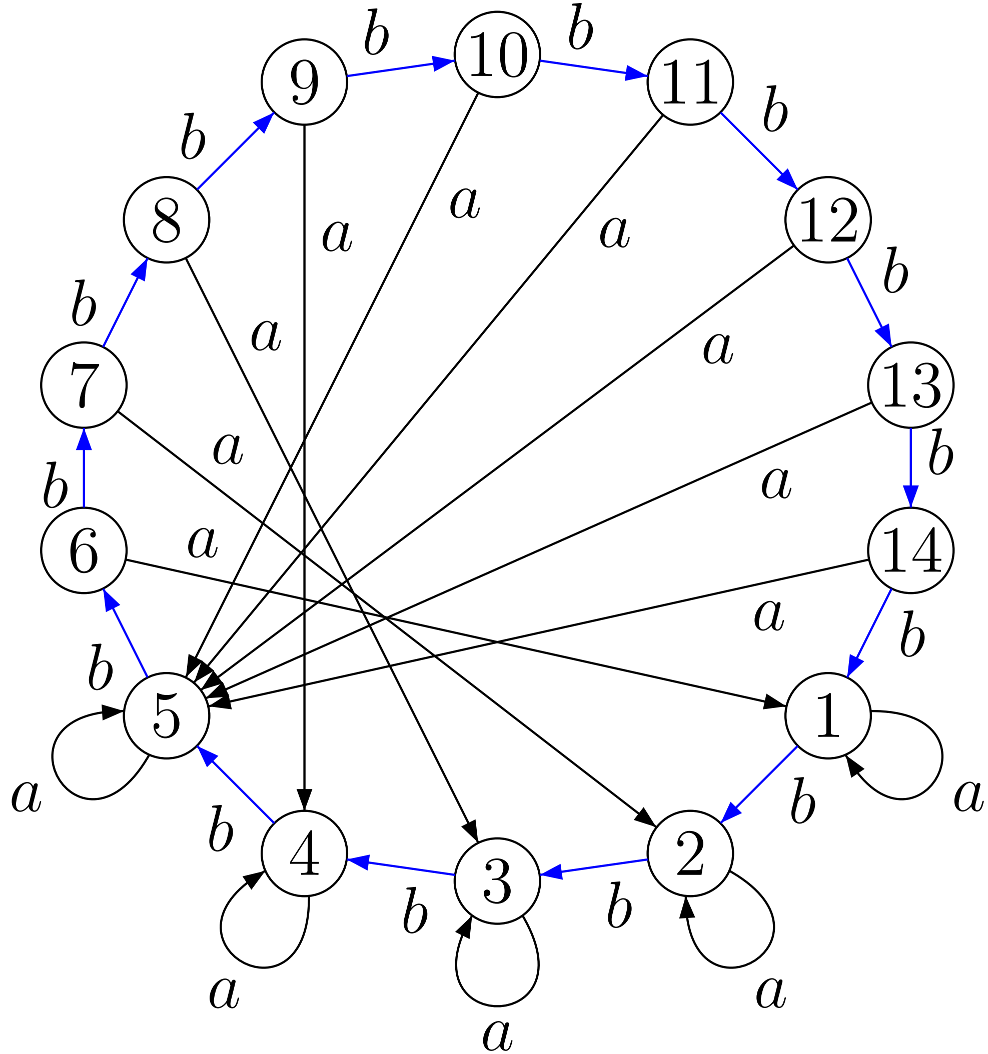
“Honest” example: “greedy” reset word (k=2)



$abbbbbbabbbbbbabbbbbbabbbbbb\textcolor{red}{a}$

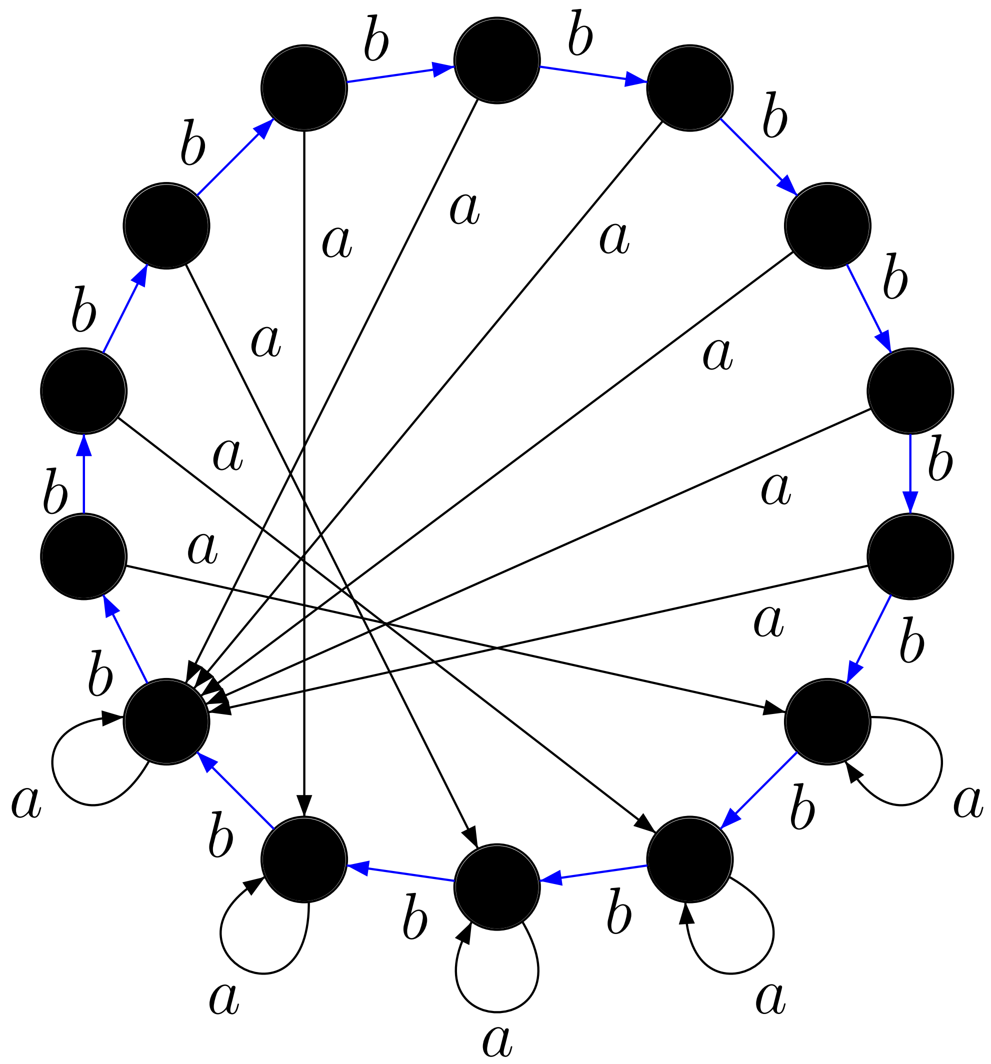
approximation ratio $\frac{29}{11}$ or $\geq \frac{n-1}{6(k-1)}$

“Honest” example: “greedy” reset word (k=3)



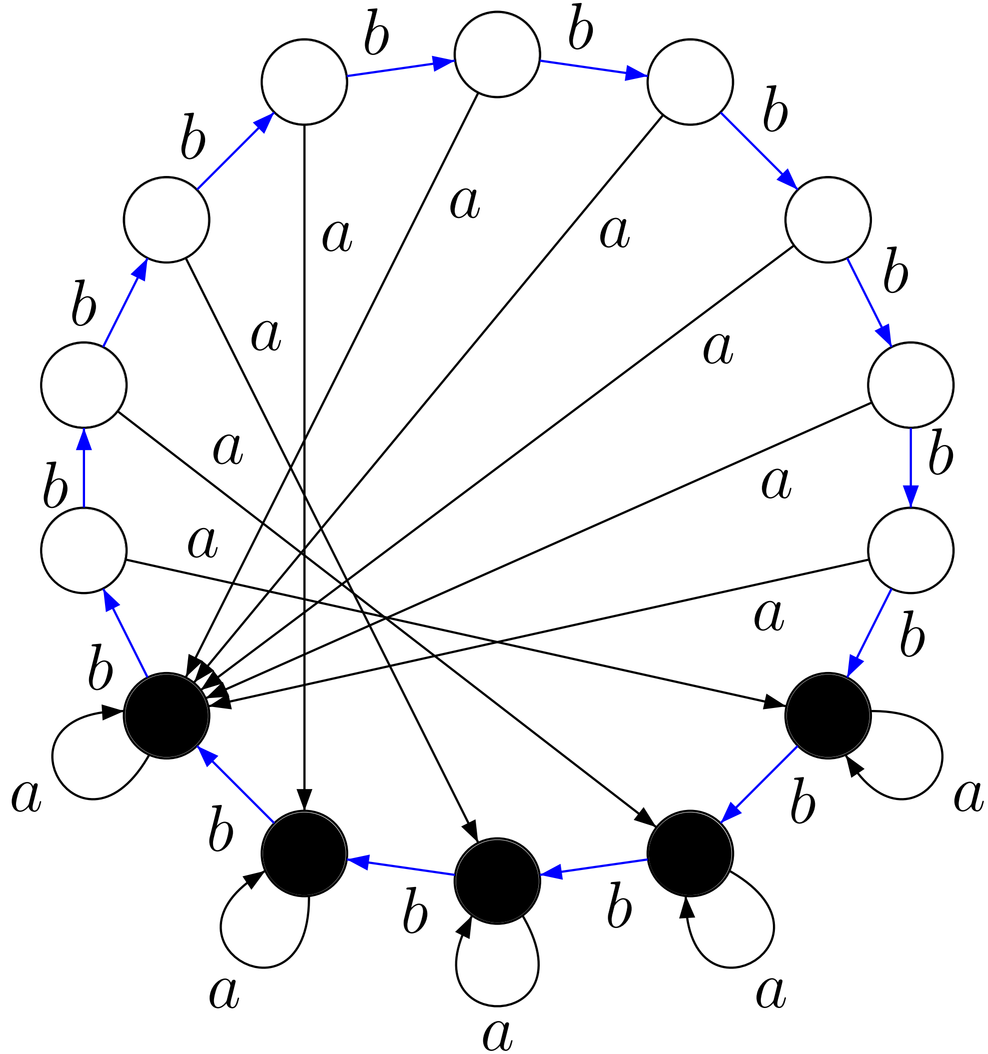
$abbbbbbbabbbbbbba$

“Honest” example: “greedy” reset word (k=3)



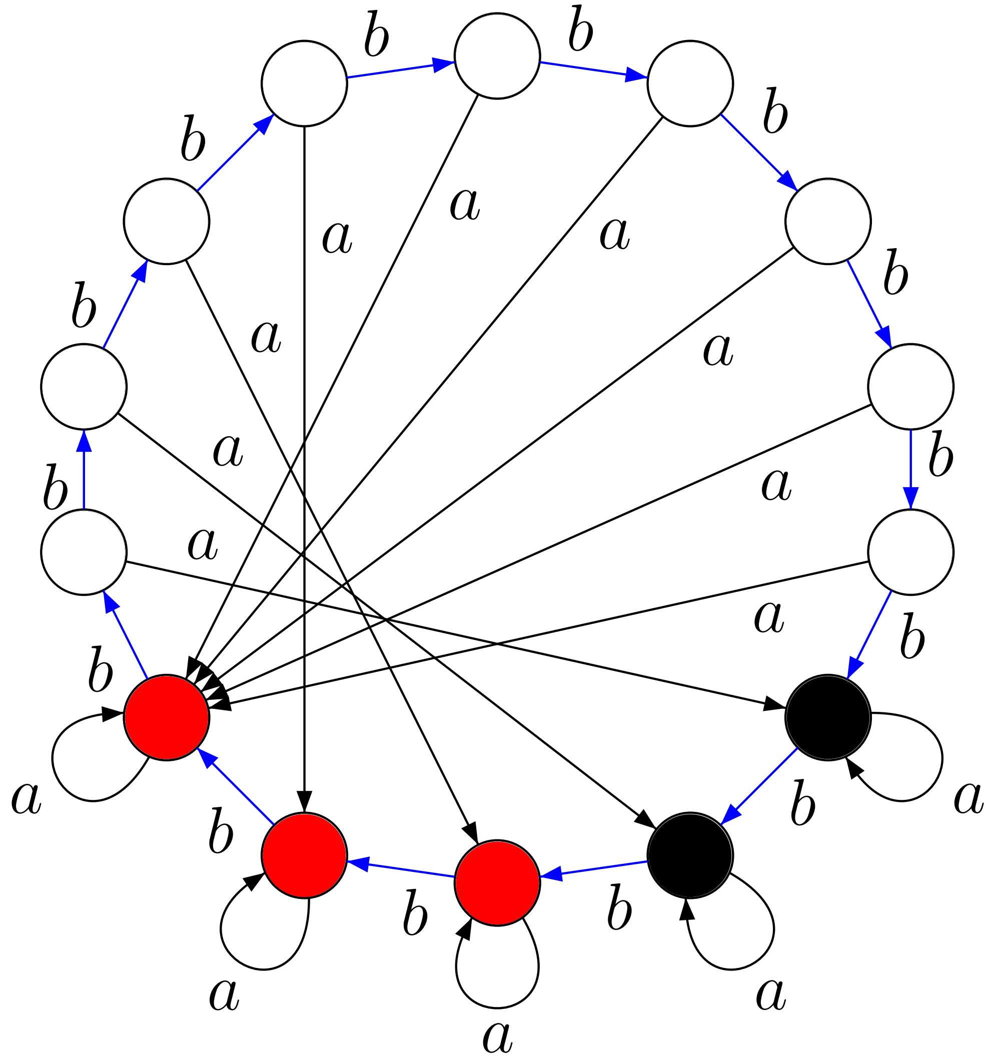
$abbbbbbbabbbbbbba$

“Honest” example: “greedy” reset word (k=3)



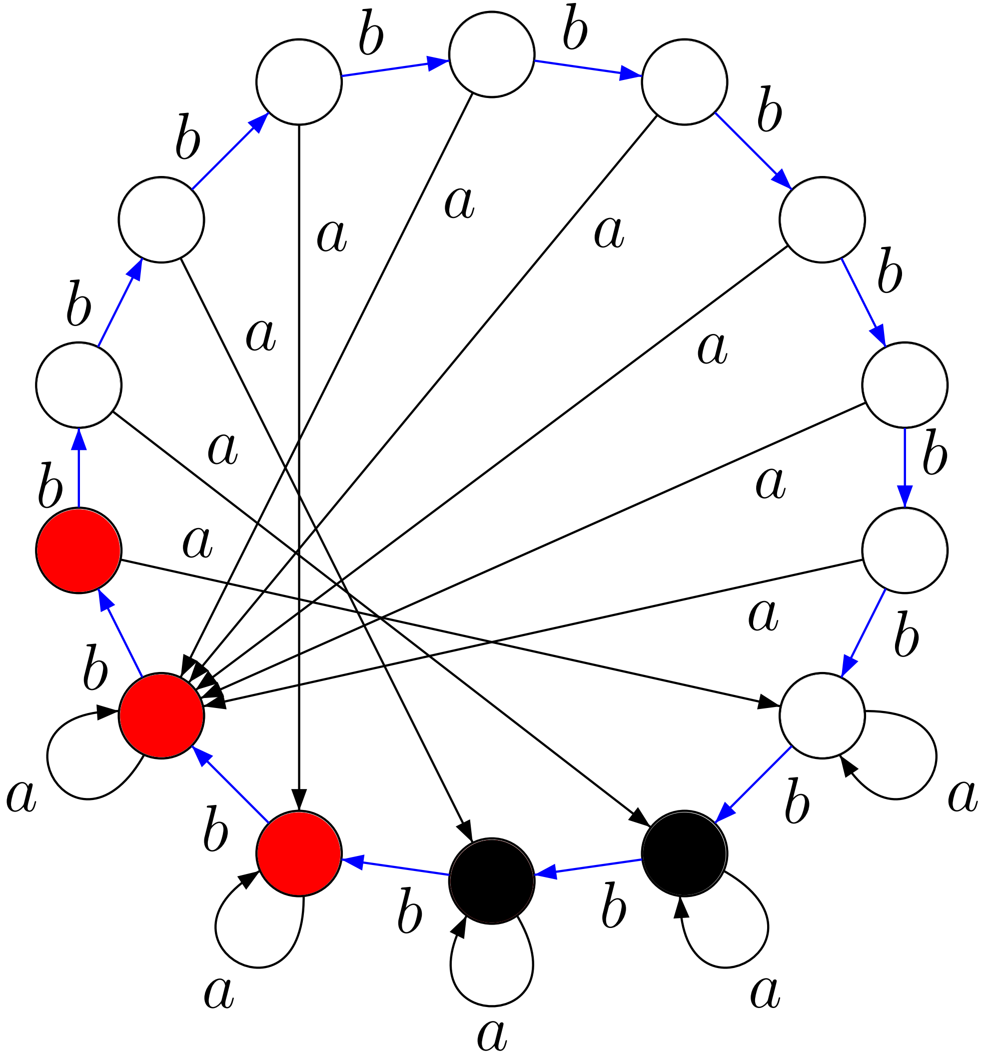
$abbbbbbabbbbbba$

“Honest” example: “greedy” reset word (k=3)



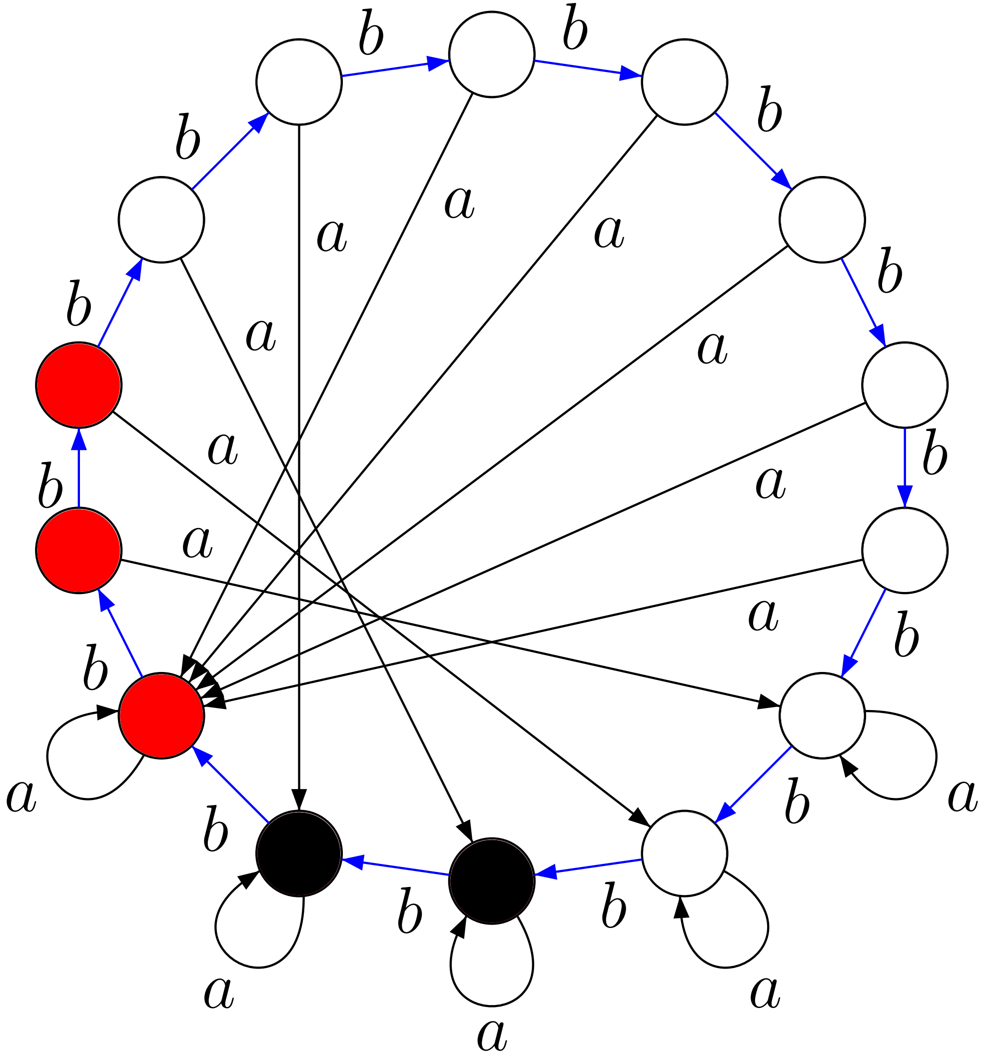
$abbbbbbbabbbbbbba$

"Honest" example: "greedy" reset word (k=3)



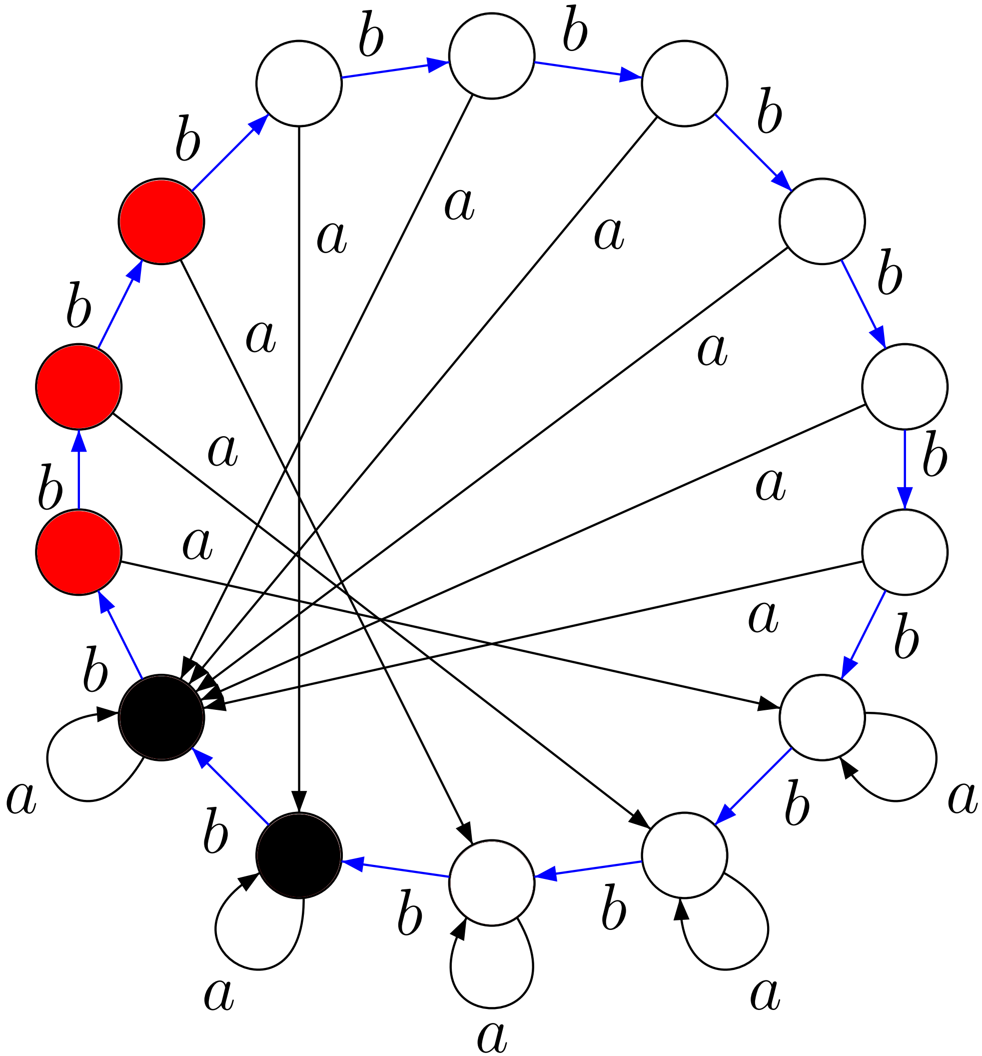
$abbbbbbabbbbbba$

“Honest” example: “greedy” reset word (k=3)



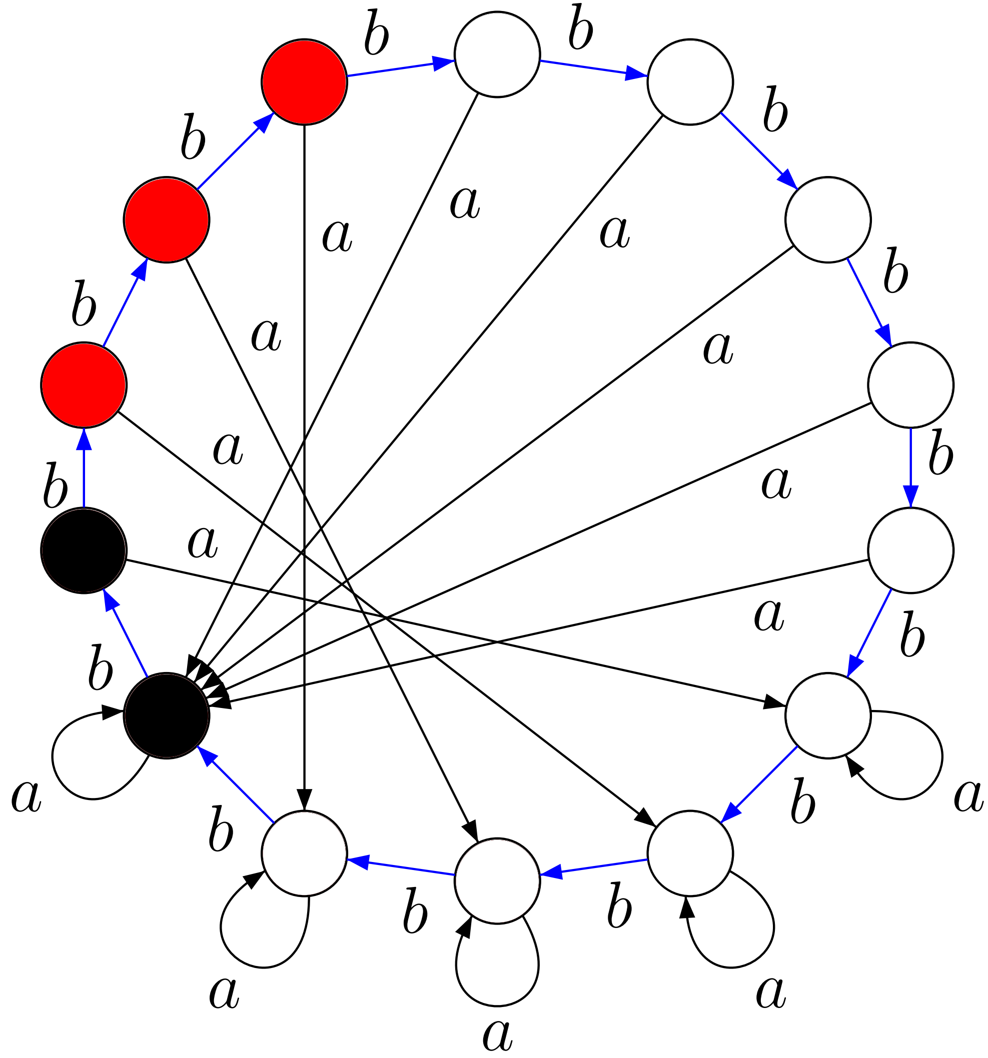
$ab\textcolor{red}{b}bbbbbabbbbbbba$

“Honest” example: “greedy” reset word (k=3)



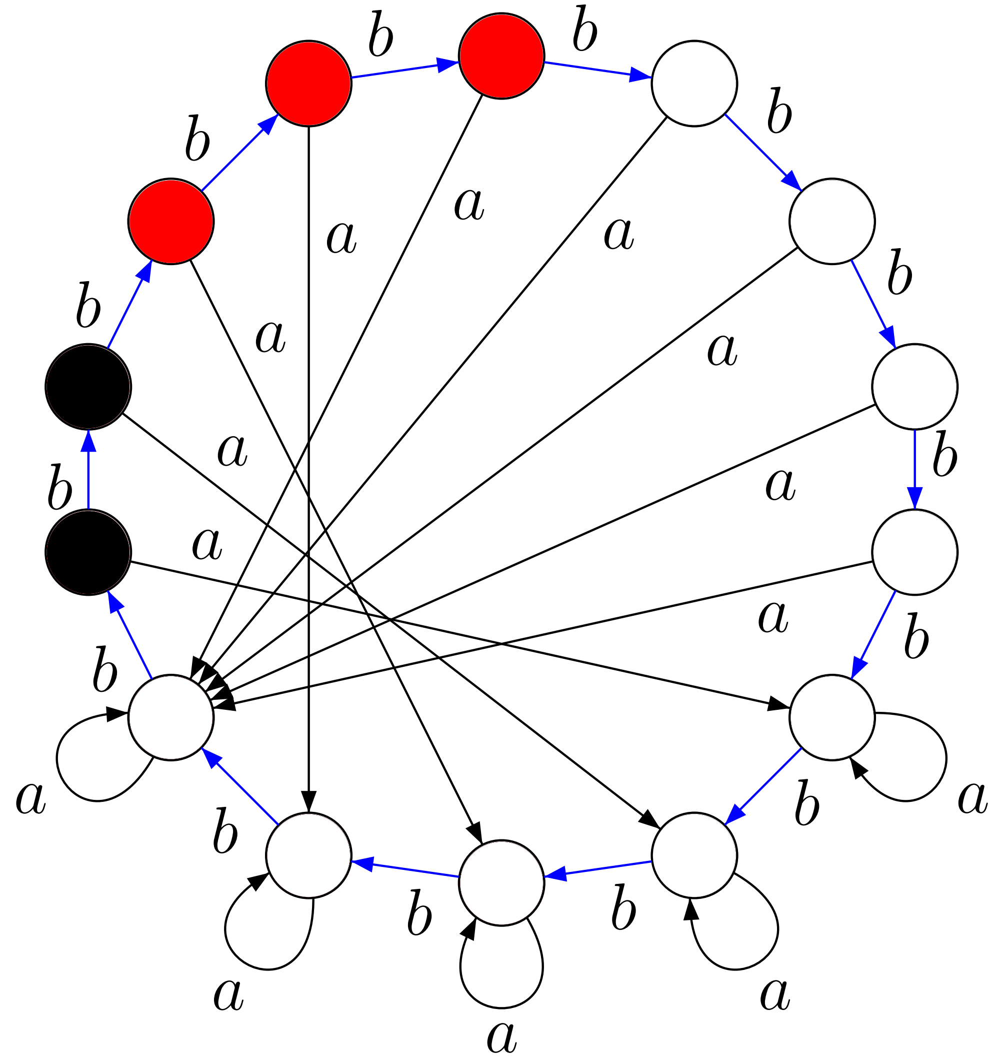
$ab b\color{red}{b}bbbbabbbbbbba$

"Honest" example: "greedy" reset word (k=3)



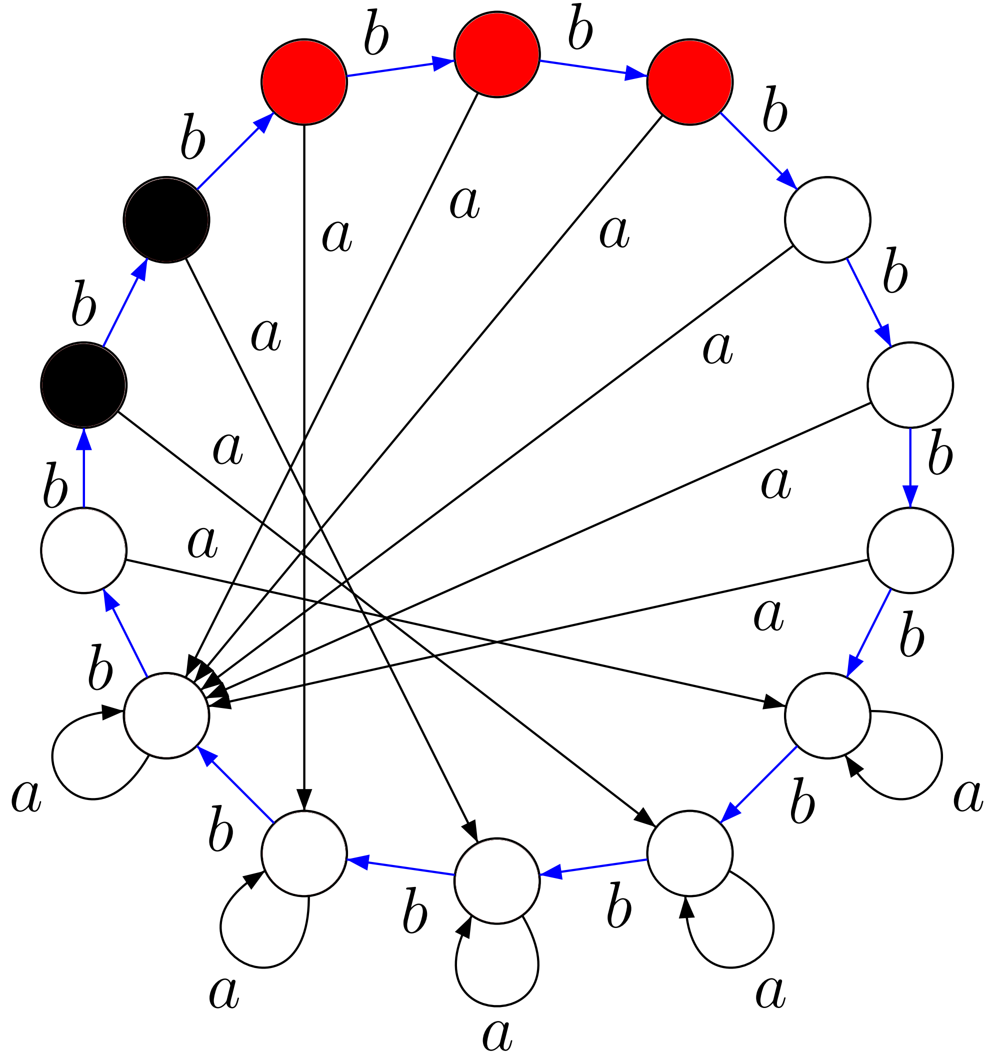
$abbb\textcolor{red}{b}bbbabbbbbba$

“Honest” example: “greedy” reset word (k=3)



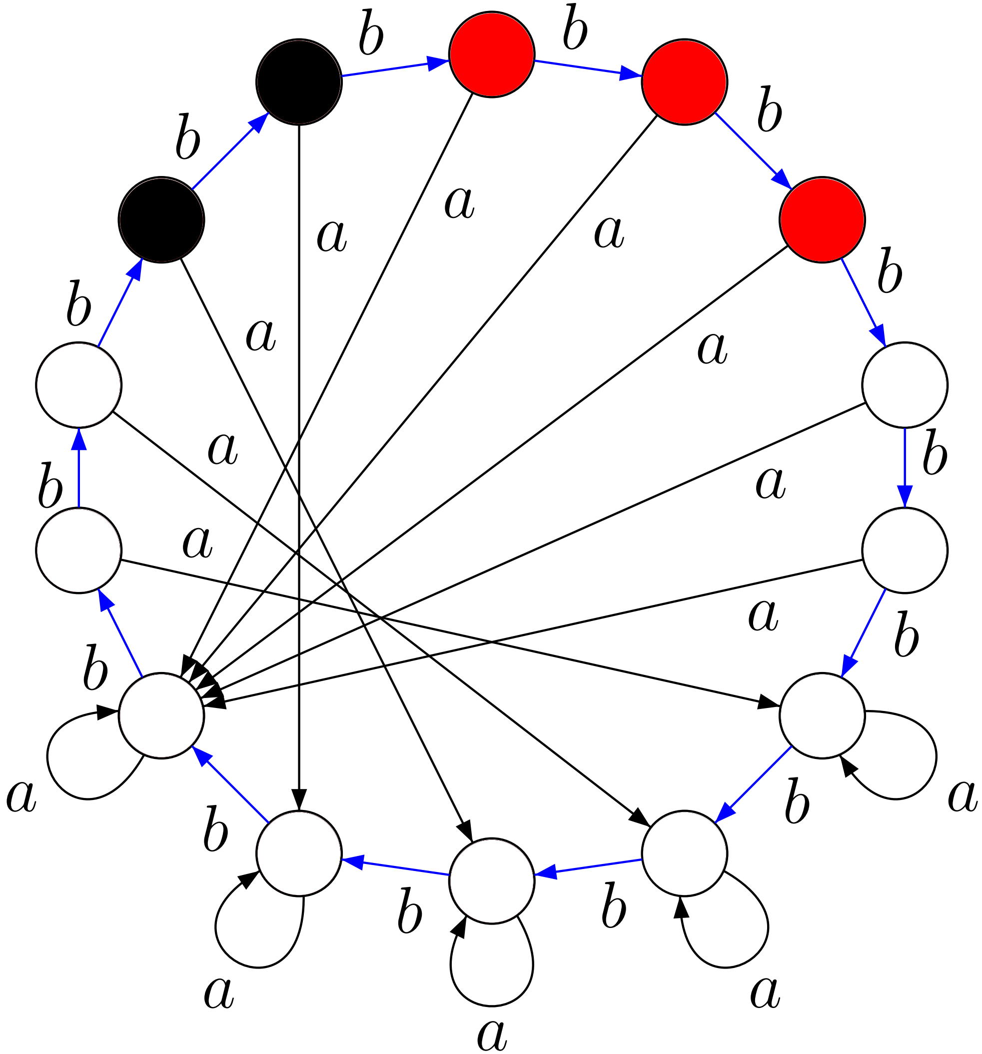
$abbbb\textcolor{red}{b}bbabbbbbba$

“Honest” example: “greedy” reset word (k=3)



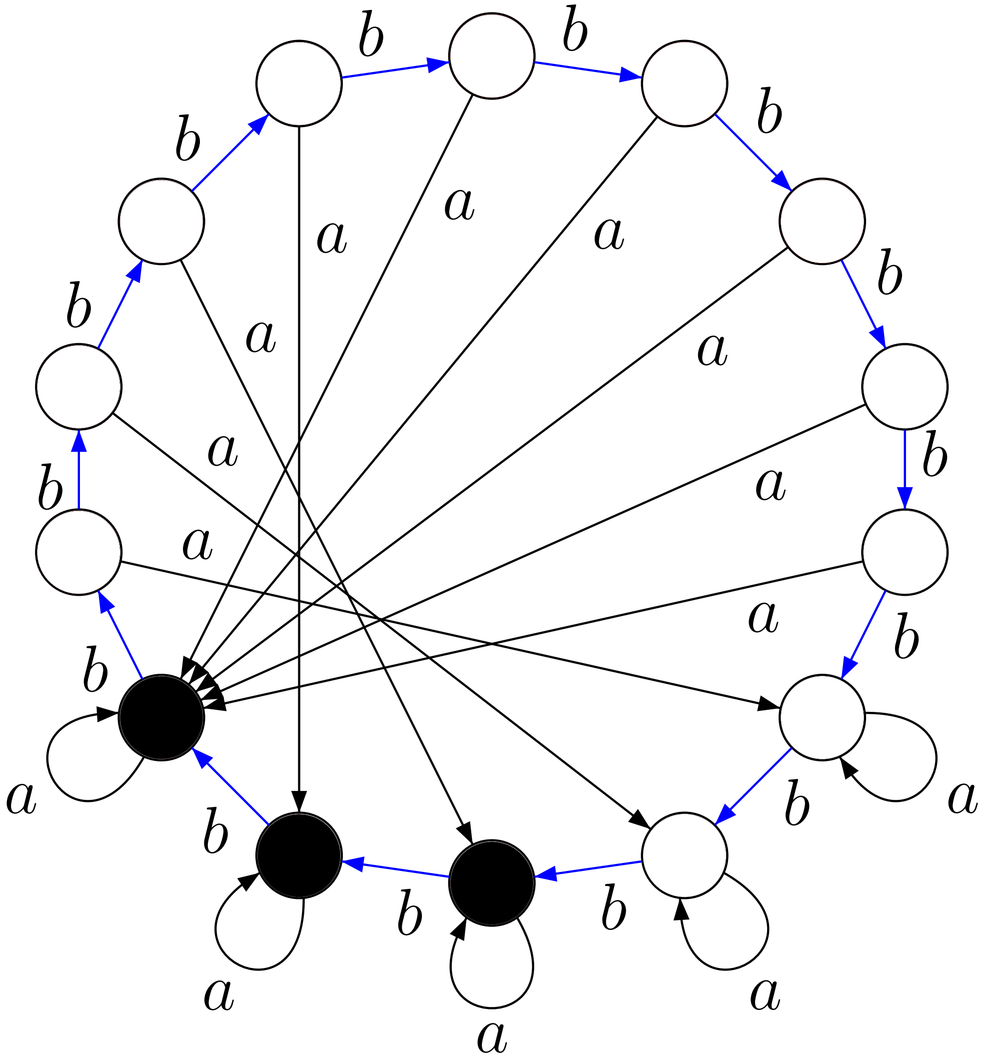
abbbbbbbabbbbbbba

“Honest” example: “greedy” reset word (k=3)



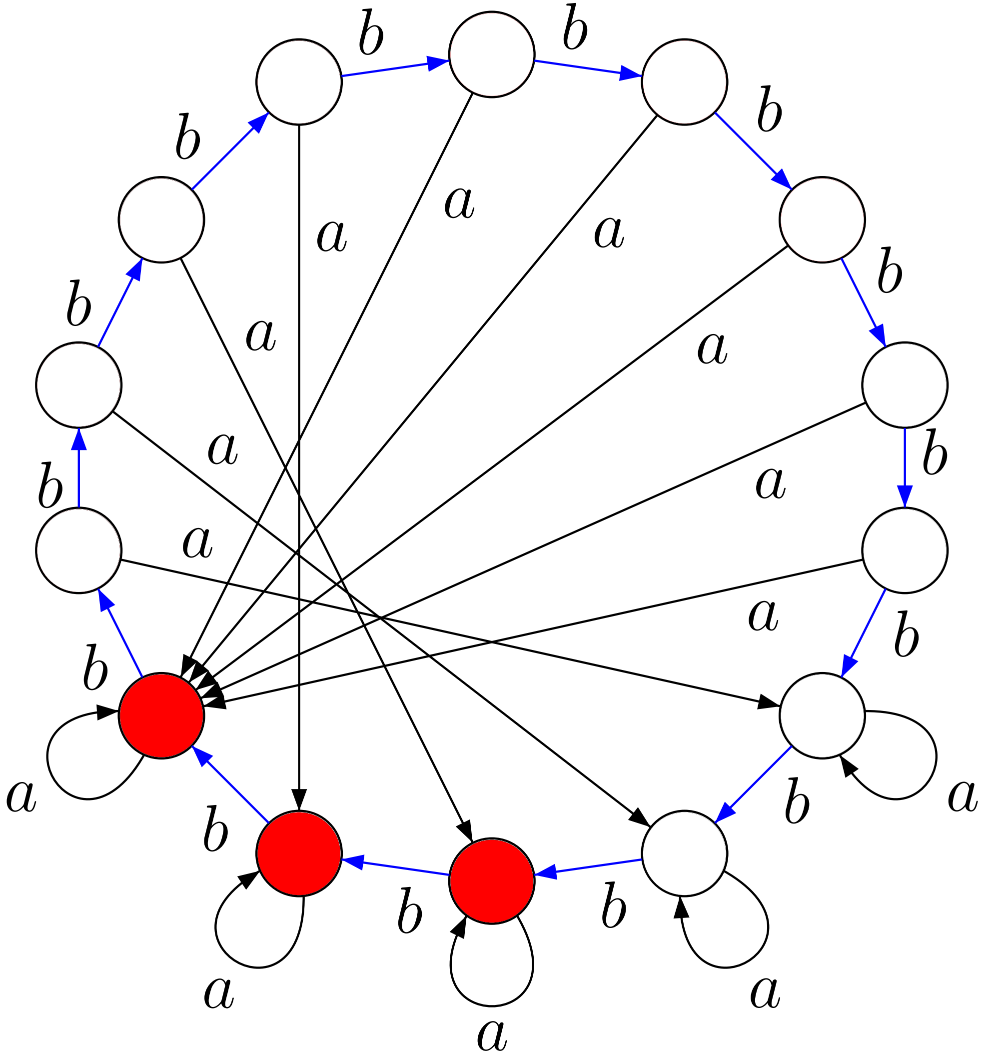
$abbbbbb\textcolor{red}{b}abbbbbbba$

“Honest” example: “greedy” reset word (k=3)



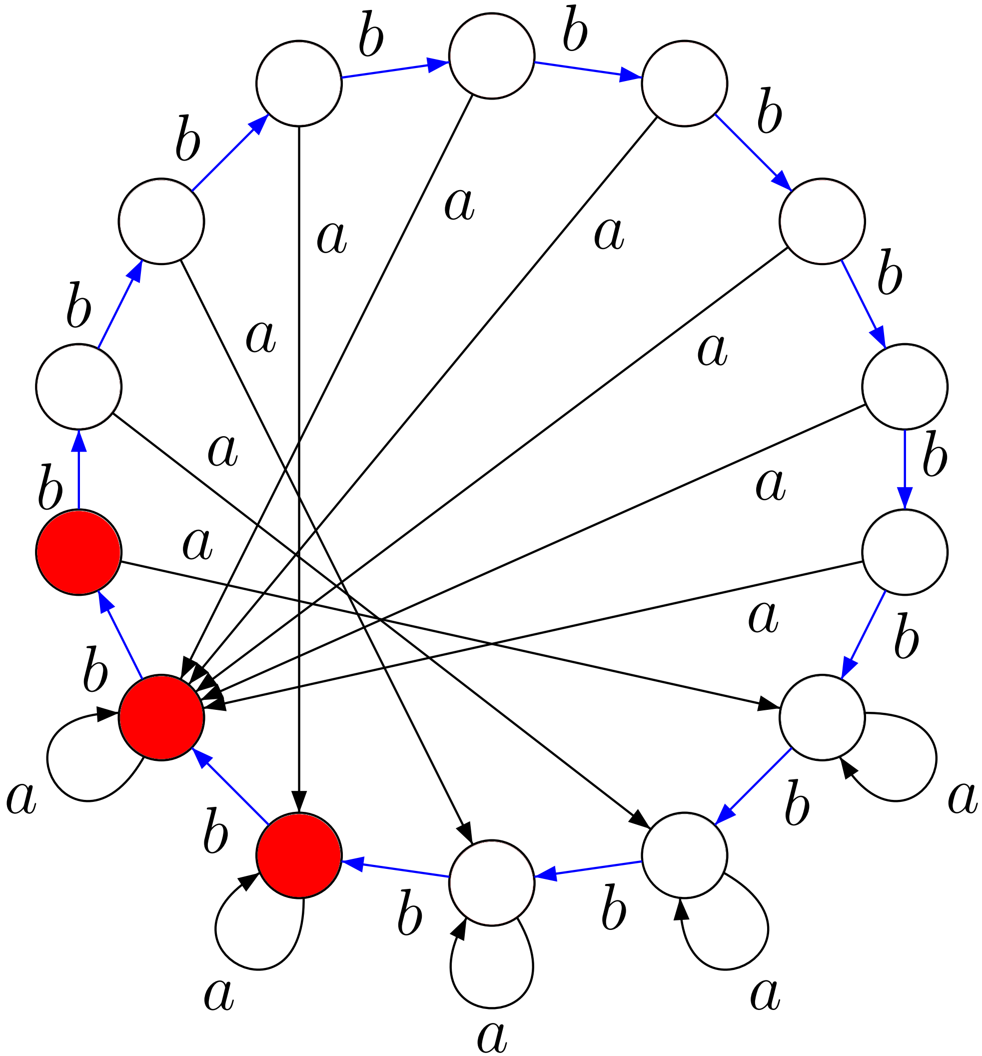
$abbbbbbbabbbbbbba$

“Honest” example: “greedy” reset word (k=3)



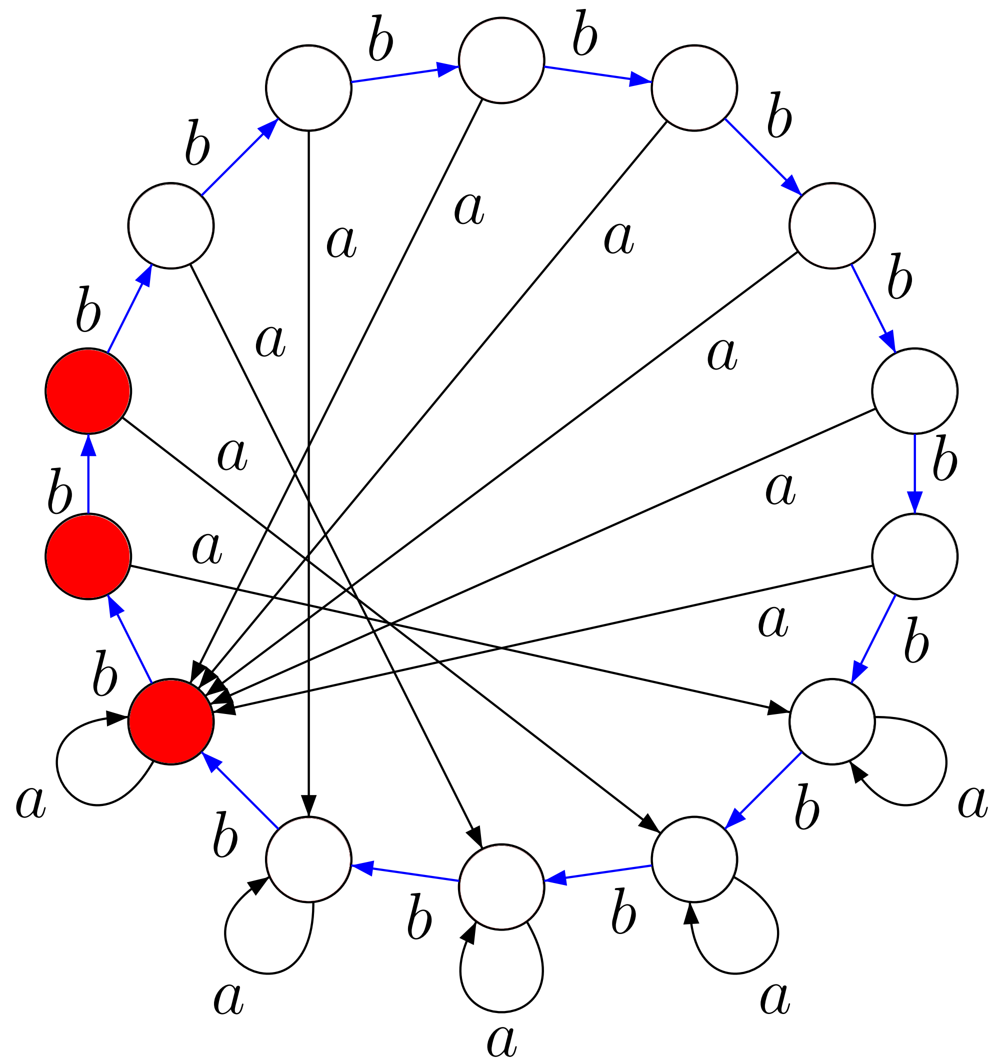
$abbbbbbbabbbbbbba$

“Honest” example: “greedy” reset word (k=3)



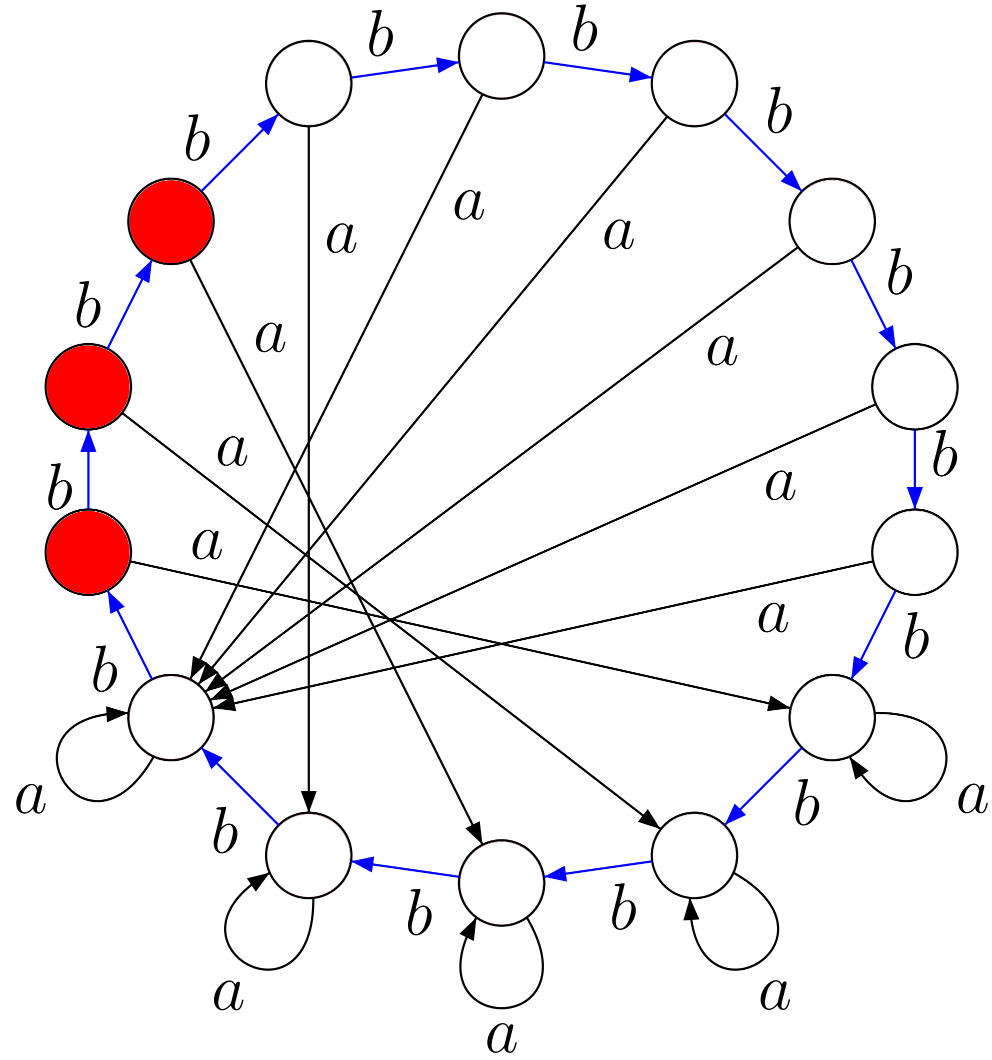
$abbbbbbbabbbbbbba$

“Honest” example: “greedy” reset word (k=3)



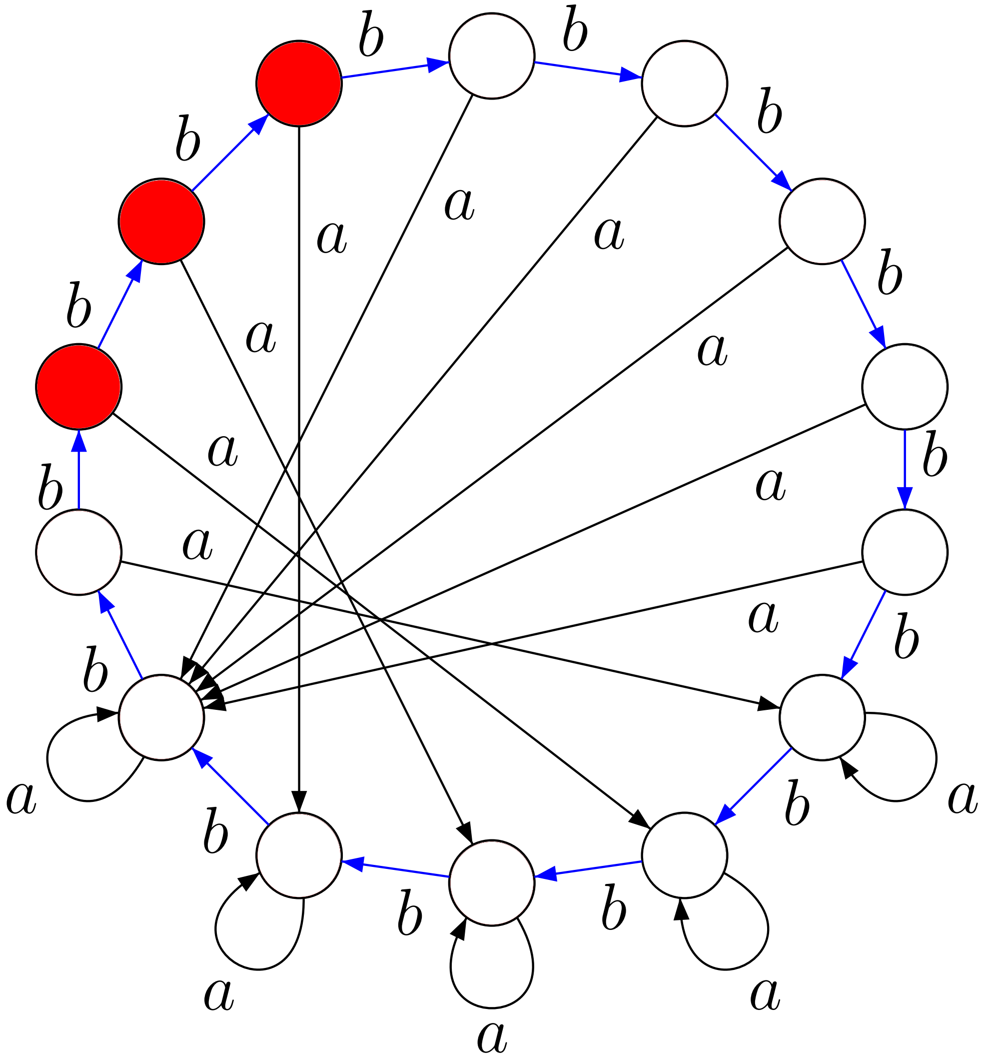
$abbbbbbbab\textcolor{red}{b}bbbbba$

“Honest” example: “greedy” reset word (k=3)



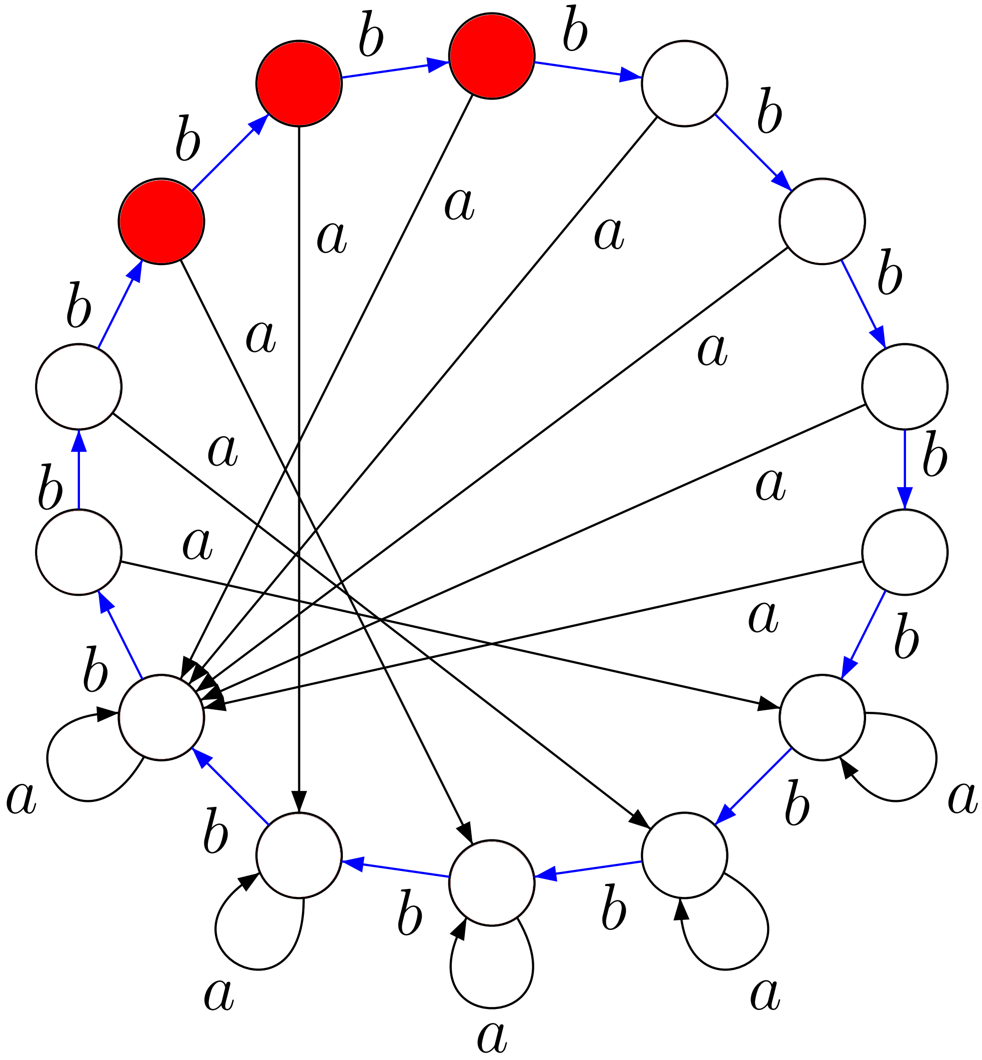
$abbbbbbbabbbbbbba$

"Honest" example: "greedy" reset word (k=3)



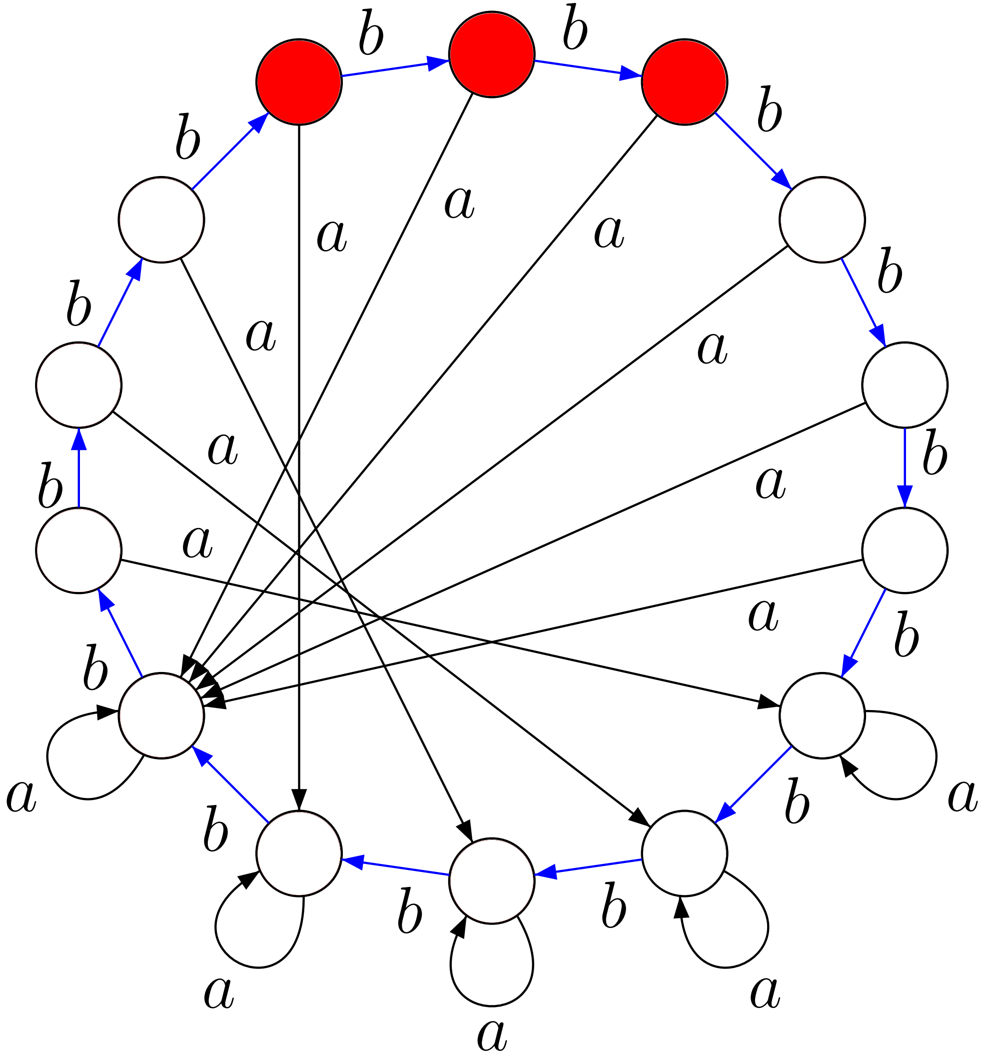
$abbbbbbbabbbbbbba$

“Honest” example: “greedy” reset word (k=3)



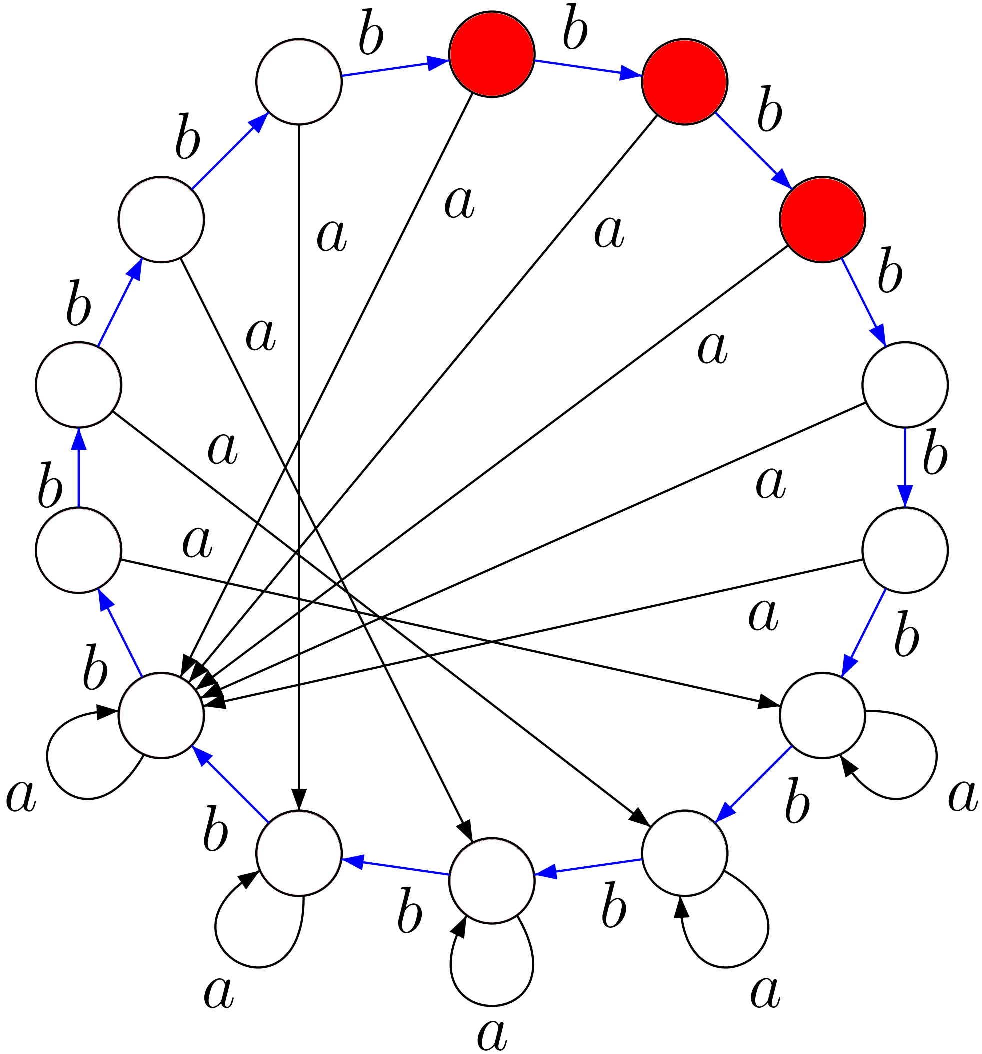
$abbbbbbbabbbbbbba$

“Honest” example: “greedy” reset word (k=3)



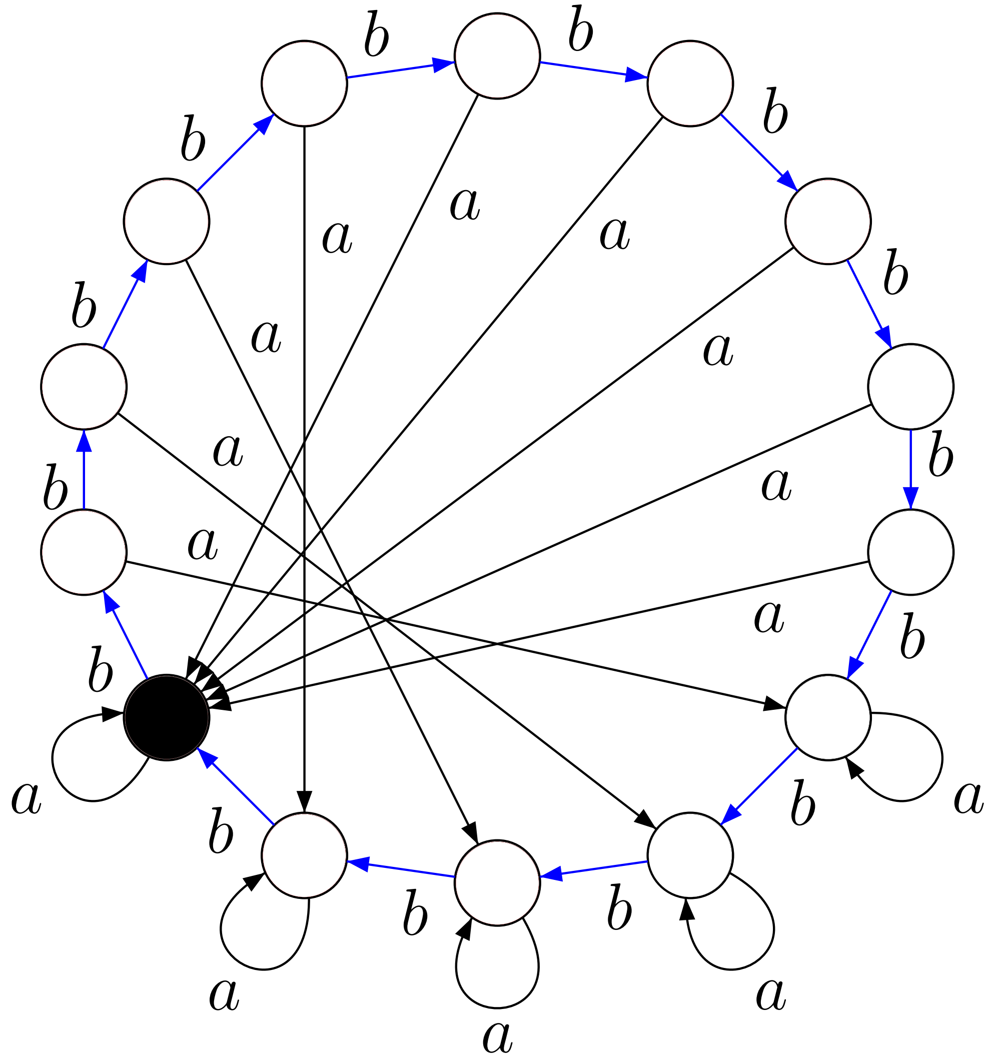
$abbbbbbbabbbbb\textcolor{red}{b}ba$

“Honest” example: “greedy” reset word (k=3)



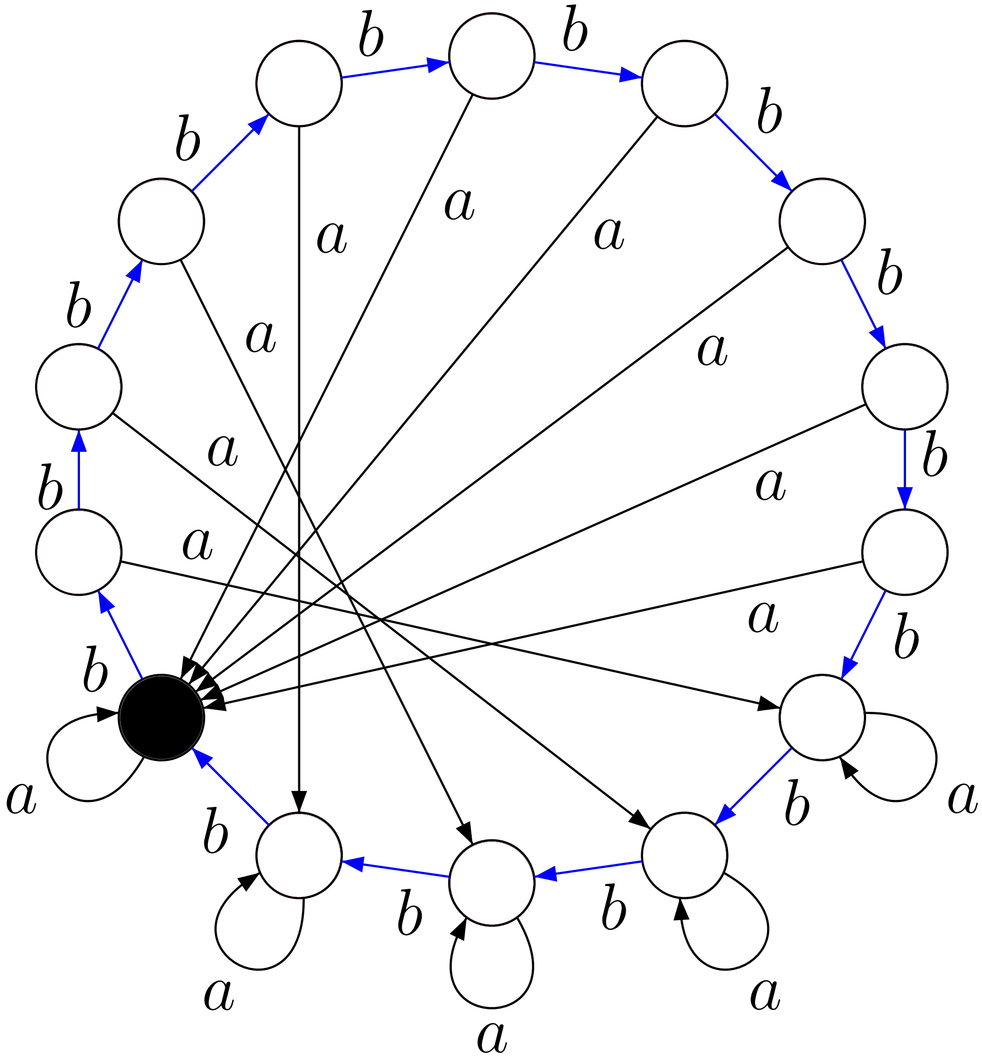
$abbbbbbabbbbbbba$

“Honest” example: “greedy” reset word (k=3)



$abbbbbbbabbbbbbba$

“Honest” example: “greedy” reset word (k=3)



$abbbbbbbabbbbbbba$

approximation ratio $\frac{17}{11}$ or $\geq \frac{n-1}{6(k-1)}$

Theorem

Approximation ratio of $SYNCH-SUBSET(k)$ with arbitrary greedy choice of word and arbitrary greedy choice of set is at least $\frac{n}{6(k-1)}$.

Theorem
Approximation ratio of $SYNCH - SUBSET(k)$ with arbitrary greedy choice of word and arbitrary greedy choice of set is at least $\frac{n}{6(k-1)}$.

Open Problem
Is there a constant C, s.t. the approximation ratio of $SYNCH - SUBSET(k)$ with arbitrary greedy choice of word and arbitrary (not only greedy) choice of set is at least $C \cdot n$.

End

Thank you.