Finitely Generated Ideal Languages and Synchronizing Automata

Marina Maslennikova
joint work with Vladimir Gusev and Elena Pribavkina

WORDS 2013
Turku, Finland, September 16-20, 2013

Institute of Mathematics and Computer Science
Ural Federal University, Ekaterinburg, Russia

Definitions

- A deterministic finite automaton (DFA) is a triple $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$.
- The transition function $\delta : Q \times \Sigma \to Q$ naturally extends to the free monoid Σ^*, this extension is still denoted by δ.
- A DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is called *synchronizing* if there exists a word w which leaves the automaton in one particular state no matter at which state in Q it is applied: $\delta(q, w) = \delta(q', w)$ for all $q, q' \in Q$.
- Any such word is said to be *reset* for the DFA $\mathscr{A}$.
- $\mathrm{Syn}(\mathscr{A})$ – the language of all reset words for a given automaton $\mathscr{A}$.

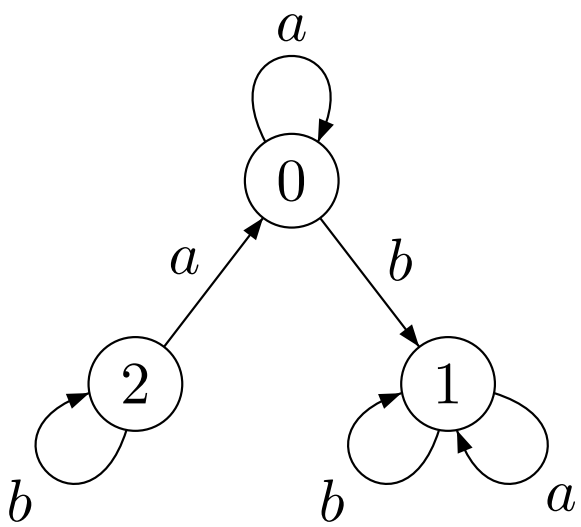
Definitions

- A deterministic finite automaton (DFA) is a triple $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$.
- The transition function $\delta : Q \times \Sigma \to Q$ naturally extends to the free monoid Σ^*, this extension is still denoted by δ.
- A DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is called *synchronizing* if there exists a word w which leaves the automaton in one particular state no matter at which state in Q it is applied: $\delta(q, w) = \delta(q', w)$ for all $q, q' \in Q$.
- Any such word is said to be *reset* for the DFA $\mathscr{A}$.
- $\mathrm{Syn}(\mathscr{A})$ – the language of all reset words for a given automaton $\mathscr{A}$.

Definitions

- A deterministic finite automaton (DFA) is a triple $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$.
- The transition function $\delta : Q \times \Sigma \to Q$ naturally extends to the free monoid Σ^*, this extension is still denoted by δ.
- A DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is called *synchronizing* if there exists a word w which leaves the automaton in one particular state no matter at which state in Q it is applied: $\delta(q, w) = \delta(q', w)$ for all $q, q' \in Q$.
- Any such word is said to be *reset* for the DFA $\mathscr{A}$.
- $\mathrm{Syn}(\mathscr{A})$ – the language of all reset words for a given automaton $\mathscr{A}$.

Example



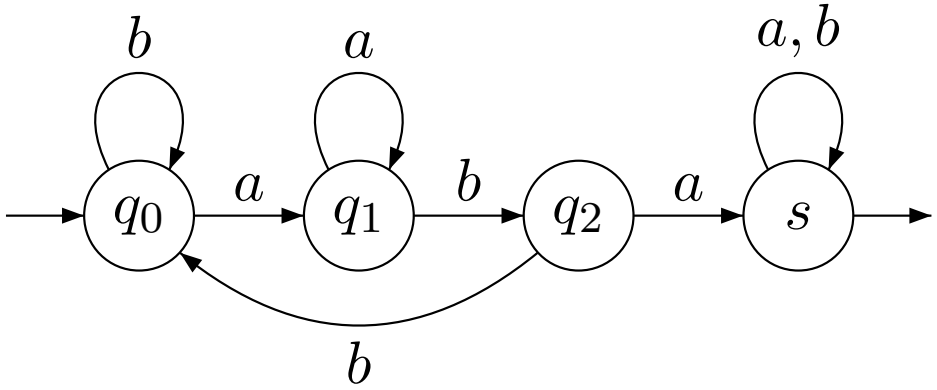
A reset word is *ab*: applying it at any state brings the automaton to the state 1.

Ideals

For every synchronizing automaton $\mathscr{A}$ over Σ it holds

$$\Sigma^* \operatorname{Syn}(\mathscr{A})\Sigma^* = \operatorname{Syn}(\mathscr{A})$$

Every regular (two-sided) ideal language is a language of reset words for some synchronizing automaton (e.g. for the minimal automaton recognizing this language).



$$L(\mathscr{A}) = I = \Sigma^* aba \Sigma^* = \operatorname{Syn}(\mathscr{A})$$

Reset complexity

- New description of ideal languages:
  given a regular ideal I and a synchronizing automaton $\mathscr{A}$ with $\mathrm{Syn}(\mathscr{A}) = I$, to check whether a word w belongs to I it is enough to apply it to each state of $\mathscr{A}$ and see whether $\delta(q, w) = \delta(q', w)$ holds for arbitrary states q, q'. This check is linear in the length of w.
- Complexity of such a representation is measured by *reset complexity*. It is the number of states in the smallest synchronizing automaton $\mathscr{A}$ with $\mathrm{Syn}(\mathscr{A}) = I$. Notation: $rc(I)$.
- In general case $rc(I) \leq sc(I)$.

Theorem [M, 2011]

For every $n \geq 3$ there are ideals I_n such that $sc(I_n) = 2^n - n$, and $rc(I_n) = n$.

Reset complexity

- New description of ideal languages:
  given a regular ideal I and a synchronizing automaton $\mathscr{A}$ with $\mathrm{Syn}(\mathscr{A}) = I$, to check whether a word w belongs to I it is enough to apply it to each state of $\mathscr{A}$ and see whether $\delta(q, w) = \delta(q', w)$ holds for arbitrary states q, q'. This check is linear in the length of w.
- Complexity of such a representation is measured by *reset complexity*. It is the number of states in the smallest synchronizing automaton $\mathscr{A}$ with $\mathrm{Syn}(\mathscr{A}) = I$. Notation: $rc(I)$.
- In general case $rc(I) \leq sc(I)$.

Theorem [M, 2011]

For every $n \geq 3$ there are ideals I_n such that $sc(I_n) = 2^n - n$, and $rc(I_n) = n$.

Reset complexity

- New description of ideal languages:
  given a regular ideal I and a synchronizing automaton $\mathscr{A}$ with $\mathrm{Syn}(\mathscr{A}) = I$, to check whether a word w belongs to I it is enough to apply it to each state of $\mathscr{A}$ and see whether $\delta(q, w) = \delta(q', w)$ holds for arbitrary states q, q'. This check is linear in the length of w.
- Complexity of such a representation is measured by *reset complexity*. It is the number of states in the smallest synchronizing automaton $\mathscr{A}$ with $\mathrm{Syn}(\mathscr{A}) = I$. Notation: $rc(I)$.
- In general case $rc(I) \leq sc(I)$.

Theorem [M, 2011]

For every $n \geq 3$ there are ideals I_n such that $sc(I_n) = 2^n - n$, and $rc(I_n) = n$.

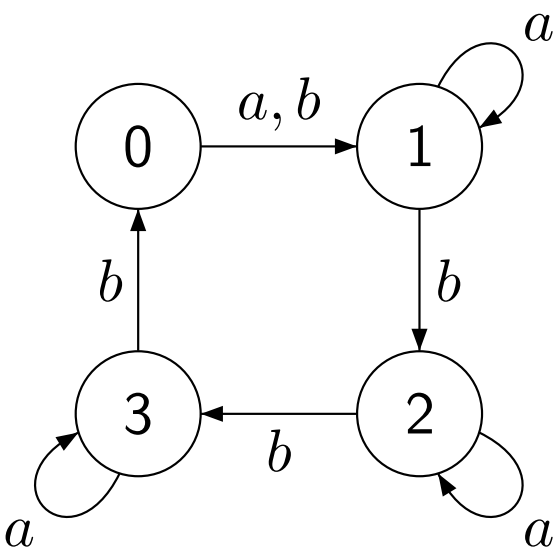
Reset complexity

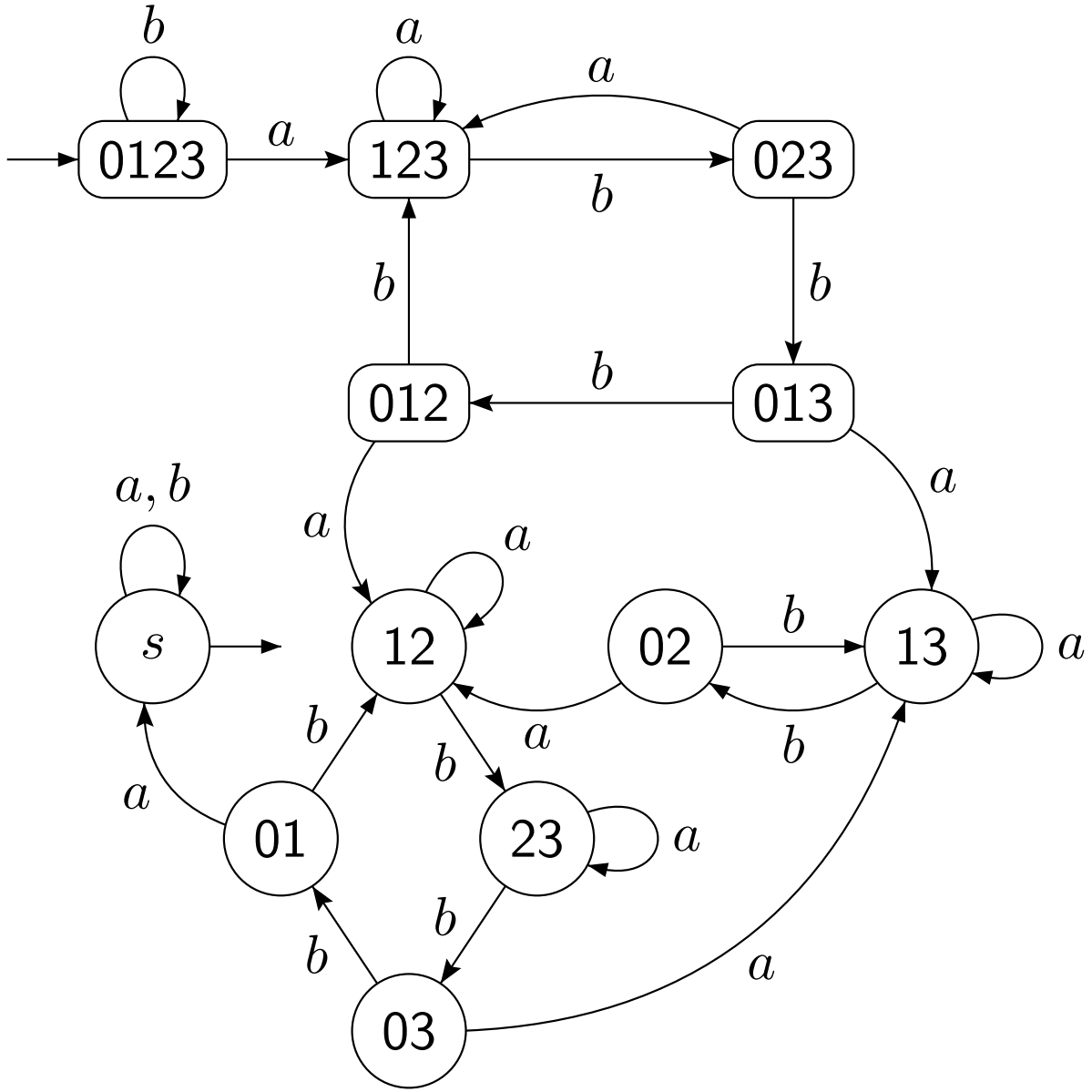
- New description of ideal languages:
  given a regular ideal I and a synchronizing automaton $\mathscr{A}$ with $\mathrm{Syn}(\mathscr{A}) = I$, to check whether a word w belongs to I it is enough to apply it to each state of $\mathscr{A}$ and see whether $\delta(q, w) = \delta(q', w)$ holds for arbitrary states q, q'. This check is linear in the length of w.
- Complexity of such a representation is measured by *reset complexity*. It is the number of states in the smallest synchronizing automaton $\mathscr{A}$ with $\mathrm{Syn}(\mathscr{A}) = I$. Notation: $rc(I)$.
- In general case $rc(I) \leq sc(I)$.

Theorem [M, 2011]

For every $n \geq 3$ there are ideals I_n such that $sc(I_n) = 2^n - n$, and $rc(I_n) = n$.

Illustration





Main problems

Let Σ be a finite alphabet with $|\Sigma| > 1$.

Question

Is it possible to construct a strongly connected synchronizing automaton for a given ideal language I yielding the minimum of the reset complexity?

In general, this is not the case.

Problem

Input: a finitely generated ideal language I over Σ, i.e. $I = \Sigma^* S \Sigma^*$ for some finite set of words S.
Task: to construct a strongly connected synchronizing automaton $\mathscr{B}$ such that $\mathrm{Syn}(\mathscr{B}) = I$.

Main problems

Let Σ be a finite alphabet with $|\Sigma| > 1$.

Question

Is it possible to construct a strongly connected synchronizing automaton for a given ideal language I yielding the minimum of the reset complexity?

In general, this is not the case.

Problem

Input: a finitely generated ideal language I over Σ, i.e. $I = \Sigma^* S \Sigma^*$ for some finite set of words S.

Task: to construct a strongly connected synchronizing automaton $\mathscr{B}$ such that $\mathrm{Syn}(\mathscr{B}) = I$.

Main problems

Let Σ be a finite alphabet with $|\Sigma| > 1$.

Question

Is it possible to construct a strongly connected synchronizing automaton for a given ideal language I yielding the minimum of the reset complexity?

In general, this is not the case.

Problem

Input: a finitely generated ideal language I over Σ, i.e. $I = \Sigma^* S \Sigma^*$ for some finite set of words S.
Task: to construct a strongly connected synchronizing automaton $\mathscr{B}$ such that $\mathrm{Syn}(\mathscr{B}) = I$.

Main problems

Let Σ be a finite alphabet with $|\Sigma| > 1$.

Question

Is it possible to construct a strongly connected synchronizing automaton for a given ideal language I yielding the minimum of the reset complexity?

In general, this is not the case.

Problem

Input: a finitely generated ideal language I over Σ, i.e. $I = \Sigma^* S \Sigma^*$ for some finite set of words S.
Task: to construct a strongly connected synchronizing automaton $\mathscr{B}$ such that $\mathrm{Syn}(\mathscr{B}) = I$.

Considered cases

- Ideal languages generated by Σ^n
- Ideal languages generated by a set of words of fixed length
- Ideal languages generated by a finite set of words
- Ideal languages generated by two words

Considered cases

- Ideal languages generated by Σ^n
- Ideal languages generated by a set of words of fixed length
- Ideal languages generated by a finite set of words
- Ideal languages generated by two words

Considered cases

- Ideal languages generated by Σ^n
- Ideal languages generated by a set of words of fixed length
- Ideal languages generated by a finite set of words
- Ideal languages generated by two words

Considered cases

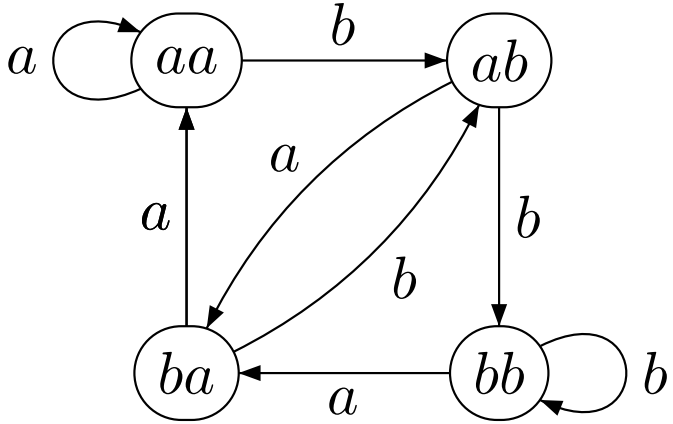
- Ideal languages generated by Σ^n
- Ideal languages generated by a set of words of fixed length
- Ideal languages generated by a finite set of words
- Ideal languages generated by two words

Ideal language generated by Σ^n

Theorem 1

Let $\Sigma = \{a, b\}$. There is unique up to isomorphism strongly connected synchronizing automaton $\mathscr{B}$ such that $\mathrm{Syn}(\mathscr{B}) = \Sigma^{\geq n}$.

The automaton $\mathscr{B}$ is actually De Brujin automaton (Rauzy graph) for the words of length n. Thus, it has 2^n states. However $rc(\Sigma^{\geq n}) = n + 1$.



Ideal language generated by a set of words of fixed length

Theorem 2

Let $U \subsetneq \Sigma^n$. There is a strongly connected synchronizing automaton $\mathscr{B}_U$ with 2^n states such that $\mathrm{Syn}(\mathscr{B}_U) = \Sigma^* U \Sigma^*$.

The construction is based on the De Bruijn automaton for the language $\Sigma^{\geq n}$. Actually, we redefine some transitions in the automaton $\mathscr{B}$ to obtain the required DFA $\mathscr{B}_U$.

Ideal language generated by a set of words of fixed length

Theorem 2

Let $U \subsetneq \Sigma^n$. There is a strongly connected synchronizing automaton $\mathscr{B}_U$ with 2^n states such that $\mathrm{Syn}(\mathscr{B}_U) = \Sigma^* U \Sigma^*$.

The construction is based on the De Bruijn automaton for the language $\Sigma^{\geq n}$. Actually, we redefine some transitions in the automaton $\mathscr{B}$ to obtain the required DFA $\mathscr{B}_U$.

Redefinition of the transitions in $\mathscr{B}$

Let we had the transition $(x, u) \xrightarrow{y} (z, v)$.

- If $uy \notin U \cup \{a^n, b^n\}$ we put $(x, u) \xrightarrow{y} (x, v)$.
- If $uy = a^n \notin U$ we put

$$(a, a^{n-1}) \xrightarrow{a} (b, a^{n-1}). \tag{1}$$

- If $uy = b^n \notin U$ we put

$$(b, b^{n-1}) \xrightarrow{b} (a, b^{n-1}). \tag{2}$$

- The other transitions remain unchanged.

Redefinition of the transitions in $\mathscr{B}$

Let we had the transition $(x, u) \xrightarrow{y} (z, v)$.

- If $uy \notin U \cup \{a^n, b^n\}$ we put $(x, u) \xrightarrow{y} (x, v)$.
- If $uy = a^n \notin U$ we put

$$(a, a^{n-1}) \xrightarrow{a} (b, a^{n-1}). \tag{1}$$

- If $uy = b^n \notin U$ we put

$$(b, b^{n-1}) \xrightarrow{b} (a, b^{n-1}). \tag{2}$$

- The other transitions remain unchanged.

Redefinition of the transitions in $\mathscr{B}$

Let we had the transition $(x, u) \xrightarrow{y} (z, v)$.

- If $uy \notin U \cup \{a^n, b^n\}$ we put $(x, u) \xrightarrow{y} (x, v)$.
- If $uy = a^n \notin U$ we put

$$(a, a^{n-1}) \xrightarrow{a} (b, a^{n-1}). \tag{1}$$

- If $uy = b^n \notin U$ we put

$$(b, b^{n-1}) \xrightarrow{b} (a, b^{n-1}). \tag{2}$$

- The other transitions remain unchanged.

Redefinition of the transitions in $\mathscr{B}$

Let we had the transition $(x,u) \xrightarrow{y} (z,v)$.

- If $uy \notin U \cup \{a^n, b^n\}$ we put $(x,u) \xrightarrow{y} (x,v)$.
- If $uy = a^n \notin U$ we put

$$(a, a^{n-1}) \xrightarrow{a} (b, a^{n-1}). \tag{1}$$

- If $uy = b^n \notin U$ we put

$$(b, b^{n-1}) \xrightarrow{b} (a, b^{n-1}). \tag{2}$$

- The other transitions remain unchanged.

Redefinition of the transitions in $\mathscr{B}$

Let we had the transition $(x, u) \xrightarrow{y} (z, v)$.

- If $uy \notin U \cup \{a^n, b^n\}$ we put $(x, u) \xrightarrow{y} (x, v)$.
- If $uy = a^n \notin U$ we put

$$(a, a^{n-1}) \xrightarrow{a} (b, a^{n-1}). \tag{1}$$

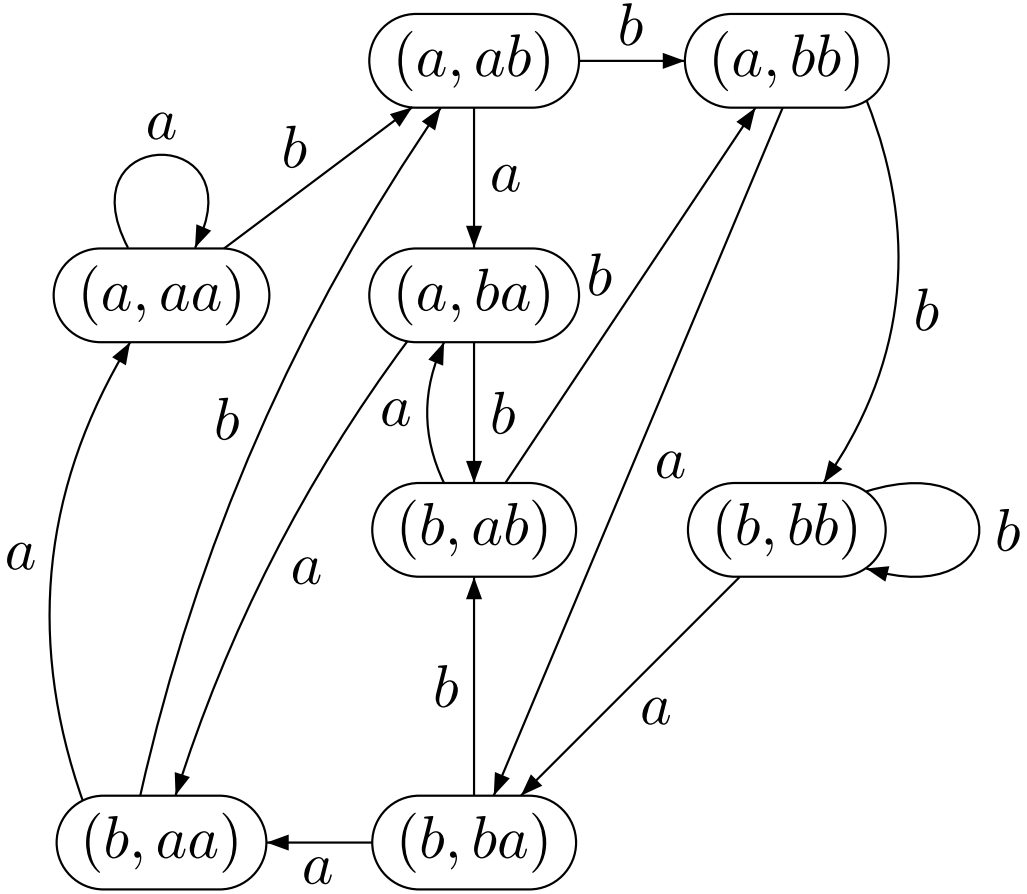
- If $uy = b^n \notin U$ we put

$$(b, b^{n-1}) \xrightarrow{b} (a, b^{n-1}). \tag{2}$$

- The other transitions remain unchanged.

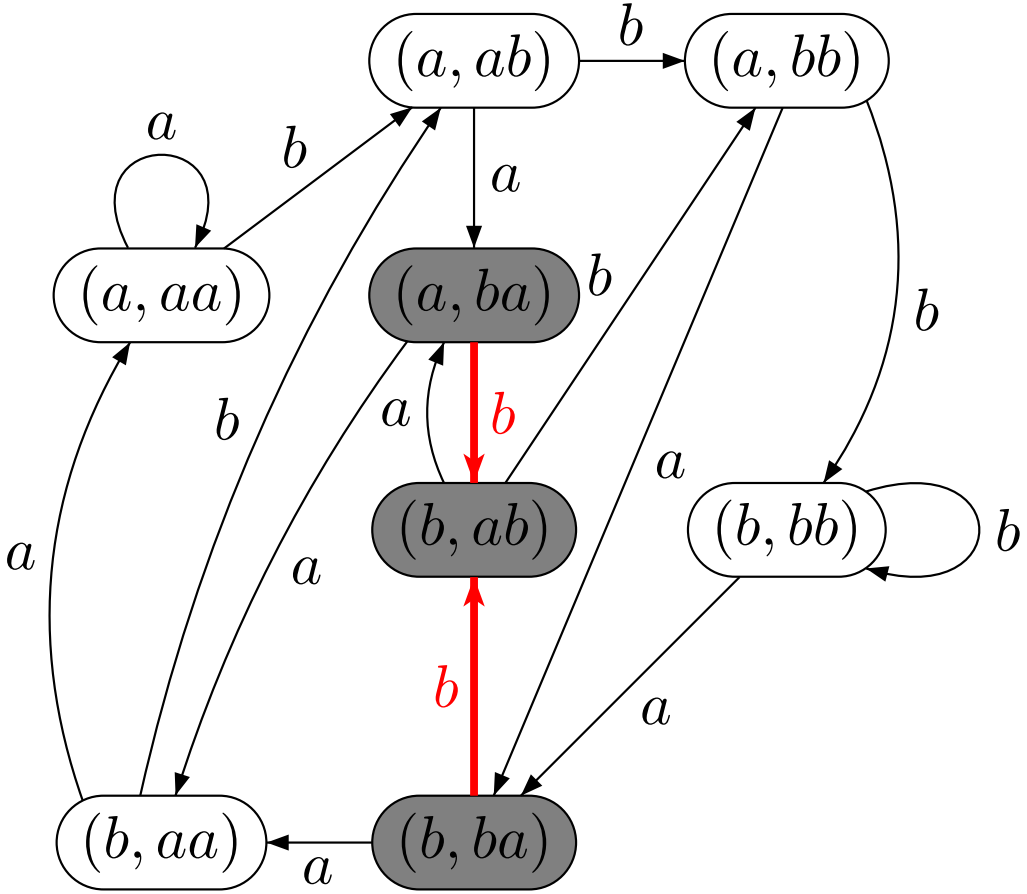
Example

Let $U = \{aaa, aab, baa, aba\}$.



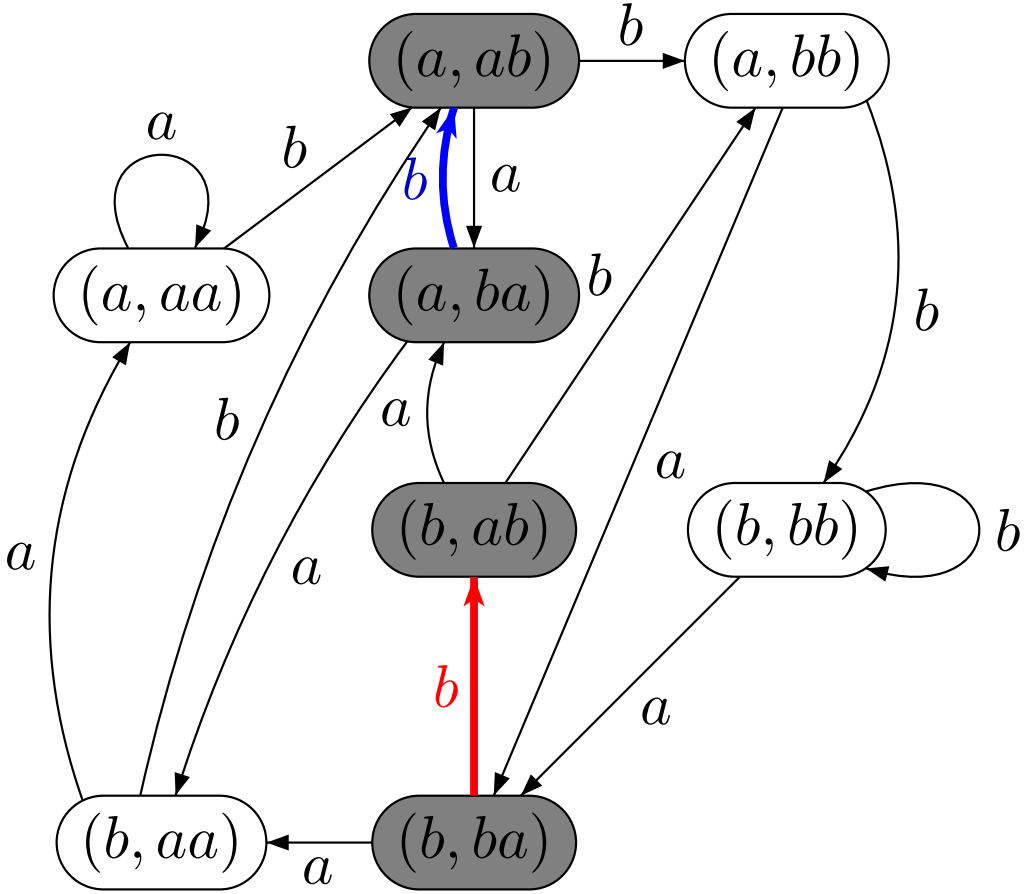
Example

Let $U = \{aaa, aab, baa, aba\}$.



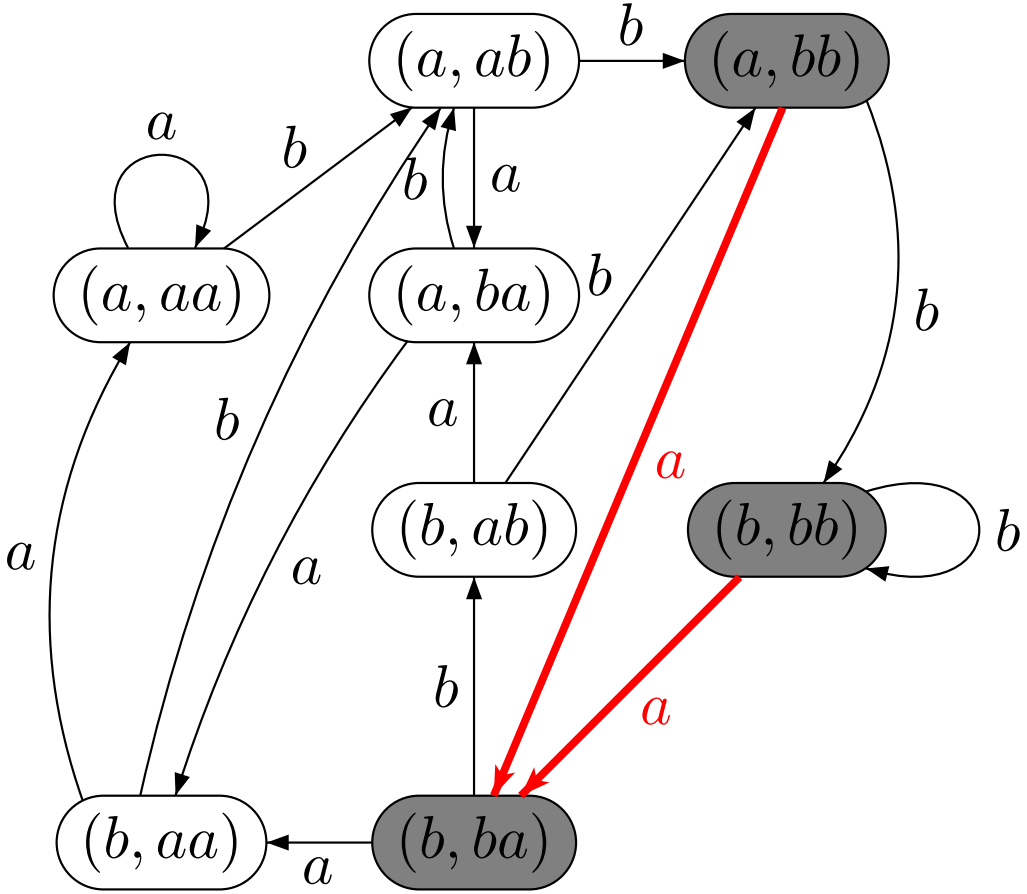
Example

Let $U = \{aaa, aab, baa, aba\}$.



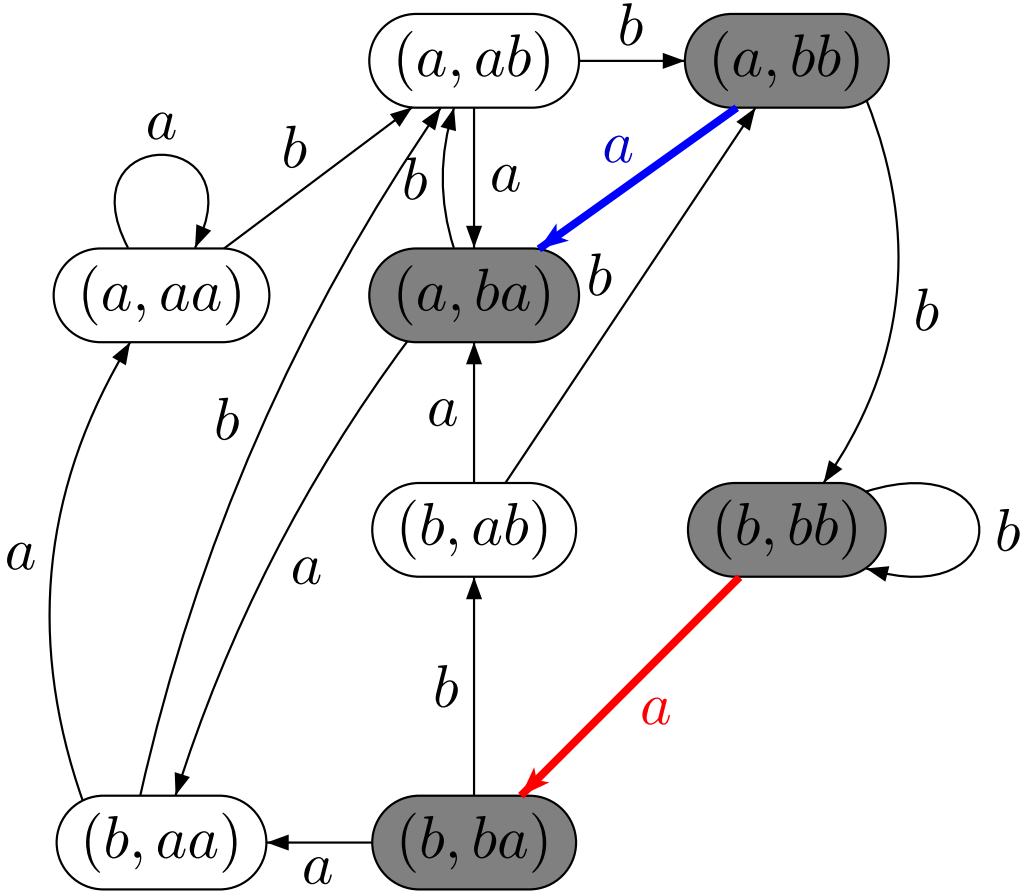
Example

Let $U = \{aaa, aab, baa, aba\}$.



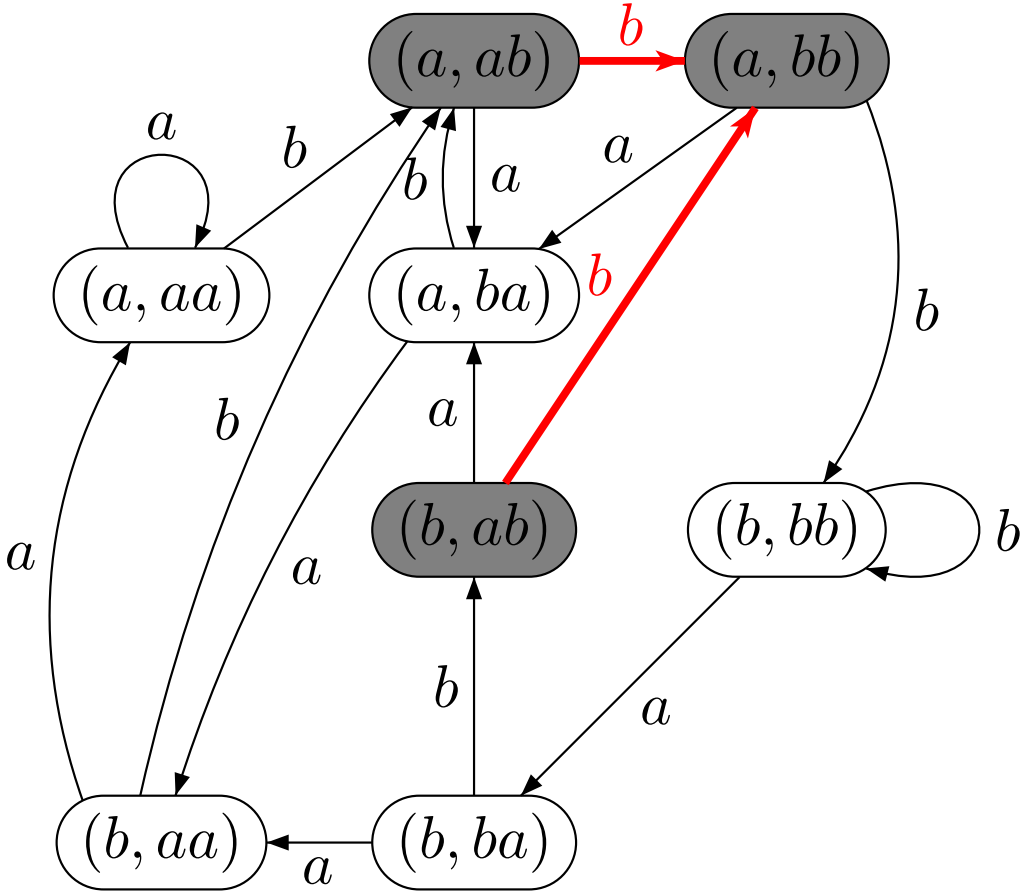
Example

Let $U = \{aaa, aab, baa, aba\}$.



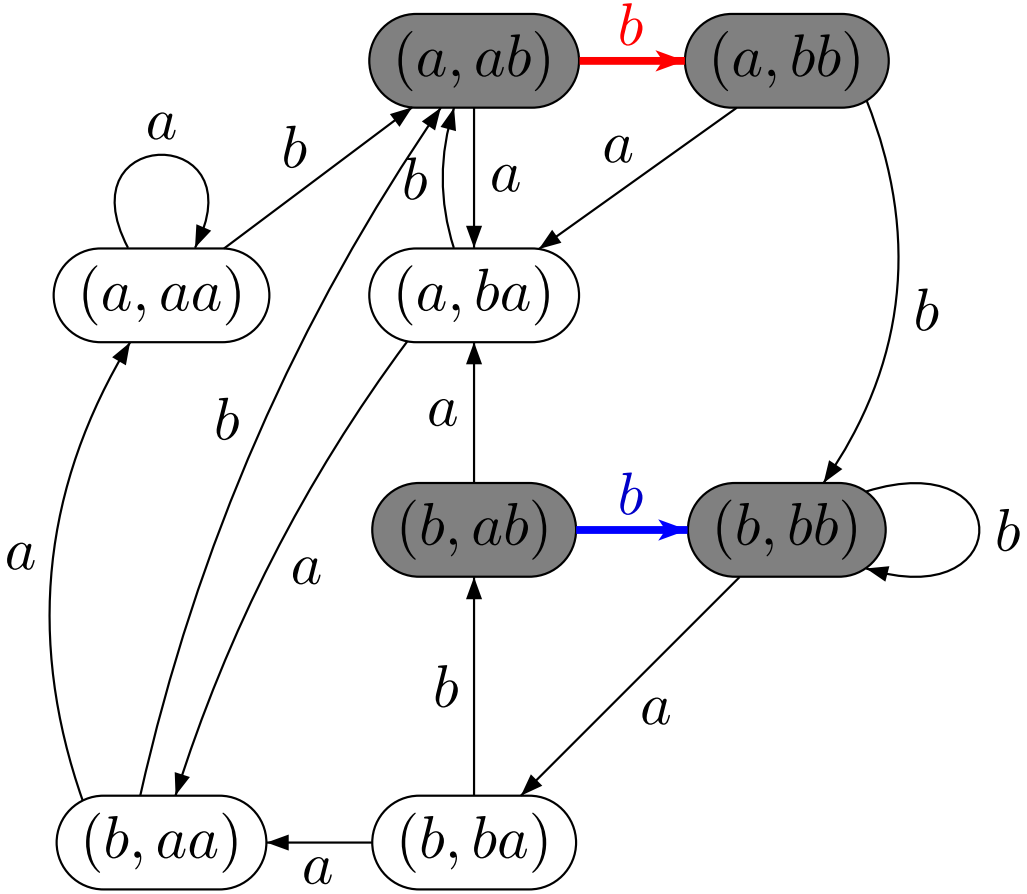
Example

Let $U = \{aaa, aab, baa, aba\}$.



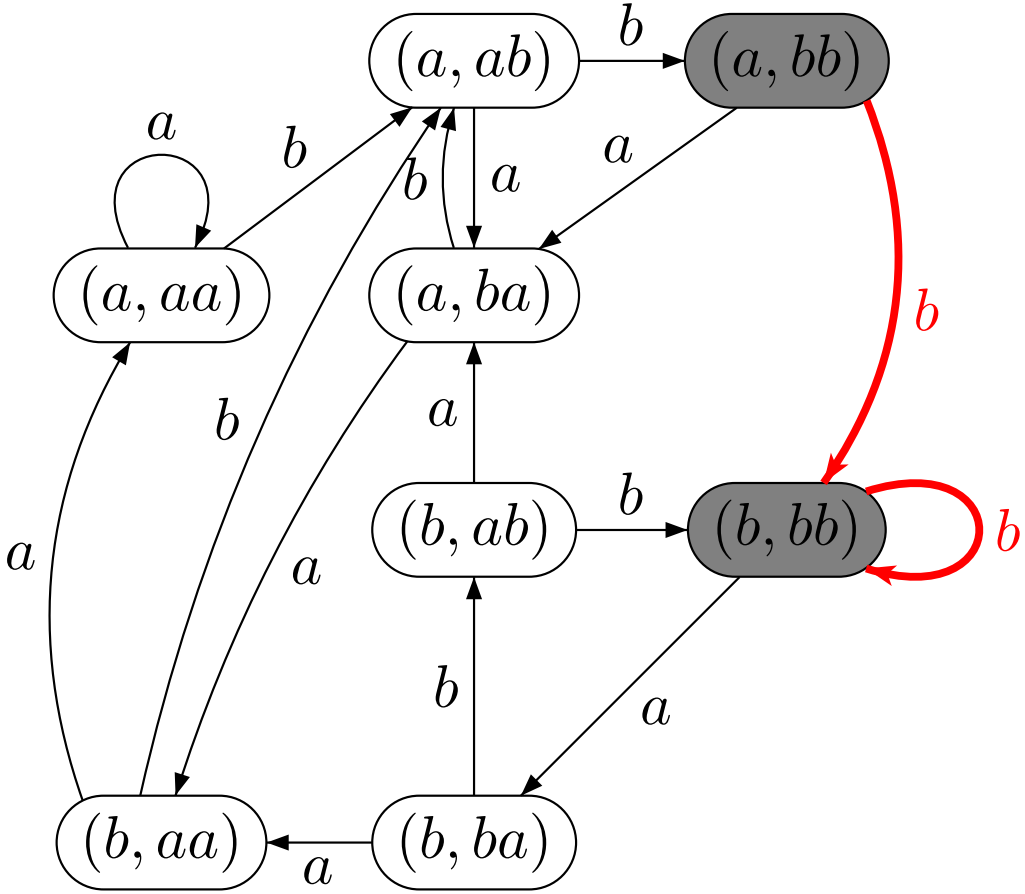
Example

Let $U = \{aaa, aab, baa, aba\}$.



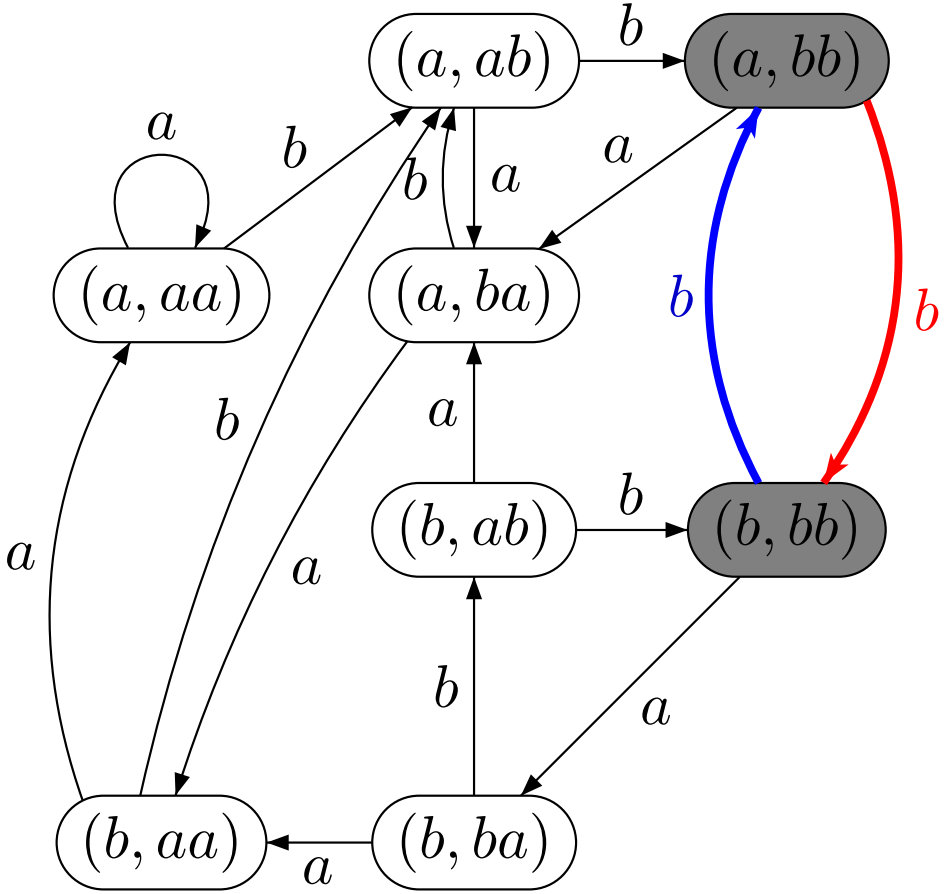
Example

Let $U = \{aaa, aab, baa, aba\}$.



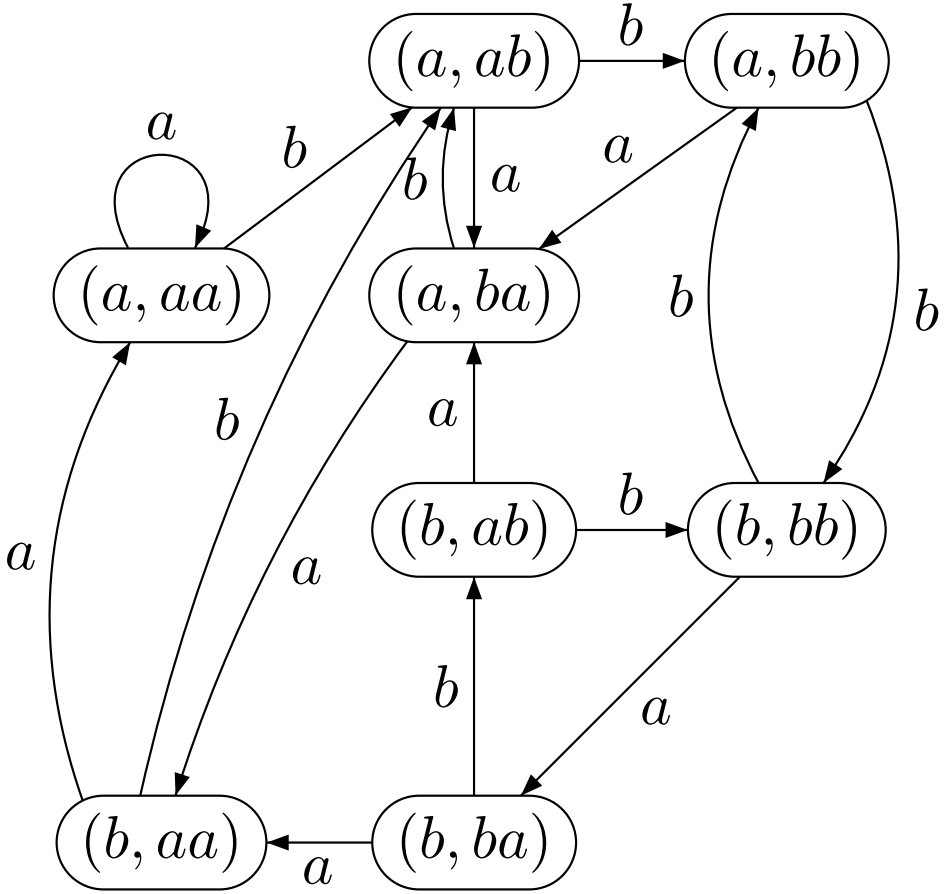
Example

Let $U = \{aaa, aab, baa, aba\}$.



Example

Let $U = \{aaa, aab, baa, aba\}$.



Ideal languages generated by a finite set of words

A finite set of words S is *anti-factorial* if no word in S is a factor of another word in S.

Theorem 3

Let S be finite and anti-factorial set of words in Σ^+. There is a strongly connected synchronizing automaton $\mathscr{C}_S$ such that $\mathrm{Syn}(\mathscr{C}_S) = \Sigma^* S \Sigma^*$. This automaton has at most 2^n states, where $n = \max\{|s| \mid s \in S\}$.

Ideal languages generated by a finite set of words

A finite set of words S is *anti-factorial* if no word in S is a factor of another word in S.

Theorem 3

Let S be finite and anti-factorial set of words in Σ^+. There is a strongly connected synchronizing automaton $\mathscr{C}_S$ such that $\mathrm{Syn}(\mathscr{C}_S) = \Sigma^* S \Sigma^*$. This automaton has at most 2^n states, where $n = \max\{|s| \mid s \in S\}$.

Algorithm. Step 1: reduction to the previous construction

- Let $T = \{w \in \Sigma^n \mid \exists s \in S, s \in \mathit{Fact}(w)\}$, where n is the maximal length of words in S.
  Example: for $S = \{aa, aba\}$ we have $T = \{aaa, aab, baa, aba\}$.
- Construct the automaton $\mathscr{B}_T$.
- The states of $\mathscr{B}_T$ are viewed not as pairs (x, u) with $x \in \Sigma, u \in \Sigma^{n-1}$, but as words xu of length n.

Algorithm. Step 1: reduction to the previous construction

- Let $T = \{w \in \Sigma^n \mid \exists s \in S, s \in \mathit{Fact}(w)\}$, where n is the maximal length of words in S.
  Example: for $S = \{aa, aba\}$ we have $T = \{aaa, aab, baa, aba\}$.
- Construct the automaton $\mathscr{B}_T$.
- The states of $\mathscr{B}_T$ are viewed not as pairs (x, u) with $x \in \Sigma, u \in \Sigma^{n-1}$, but as words xu of length n.

Algorithm. Step 1: reduction to the previous construction

- Let $T = \{w \in \Sigma^n \mid \exists s \in S, s \in \mathit{Fact}(w)\}$, where n is the maximal length of words in S.
  Example: for $S = \{aa, aba\}$ we have $T = \{aaa, aab, baa, aba\}$.
- Construct the automaton $\mathscr{B}_T$.
- The states of $\mathscr{B}_T$ are viewed not as pairs (x, u) with $x \in \Sigma, u \in \Sigma^{n-1}$, but as words xu of length n.

Algorithm. Step 1: reduction to the previous construction

- Let $T = \{w \in \Sigma^n \mid \exists s \in S, s \in \mathit{Fact}(w)\}$, where n is the maximal length of words in S.
  Example: for $S = \{aa, aba\}$ we have
  $T = \{aaa, aab, baa, aba\}$.
- Construct the automaton $\mathscr{B}_T$.
- The states of $\mathscr{B}_T$ are viewed not as pairs (x, u) with $x \in \Sigma, u \in \Sigma^{n-1}$, but as words xu of length n.

Algorithm. Step 2: factor automaton $\mathscr{B}_T/\simeq$

- Every word in T can be uniquely factorized as usv, where $s \in S$, $u, v \in \Sigma^*$ and sv does not contain factors in S except s.
- Define congruence $\simeq$: for $w, w' \in T$ $w \simeq w'$ iff $w = usv$ and $w' = u'sv$ for some $s \in S$, $u, u', v \in \Sigma^*$; every $w \notin T$ forms its own class.
- Construct the factor automaton $\mathscr{B}_T/\simeq$.

Algorithm. Step 2: factor automaton $\mathscr{B}_T/\simeq$

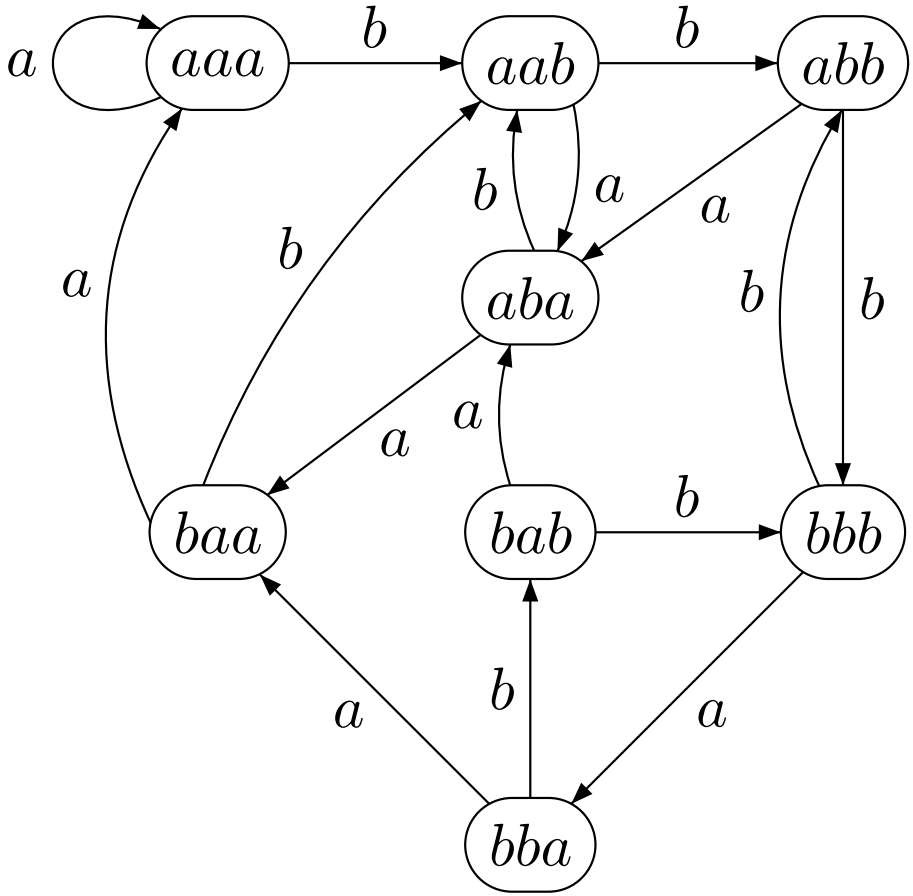
- Every word in T can be uniquely factorized as usv, where $s \in S$, $u, v \in \Sigma^*$ and sv does not contain factors in S except s.
- Define congruence $\simeq$: for $w, w' \in T$ $w \simeq w'$ iff $w = usv$ and $w' = u'sv$ for some $s \in S$, $u, u', v \in \Sigma^*$; every $w \notin T$ forms its own class.
- Construct the factor automaton $\mathscr{B}_T/\simeq$.

Algorithm. Step 2: factor automaton $\mathscr{B}_T/\simeq$

- Every word in T can be uniquely factorized as usv, where $s \in S$, $u, v \in \Sigma^*$ and sv does not contain factors in S except s.
- Define congruence $\simeq$: for $w, w' \in T$ $w \simeq w'$ iff $w = usv$ and $w' = u'sv$ for some $s \in S$, $u, u', v \in \Sigma^*$; every $w \notin T$ forms its own class.
- Construct the factor automaton $\mathscr{B}_T/\simeq$.

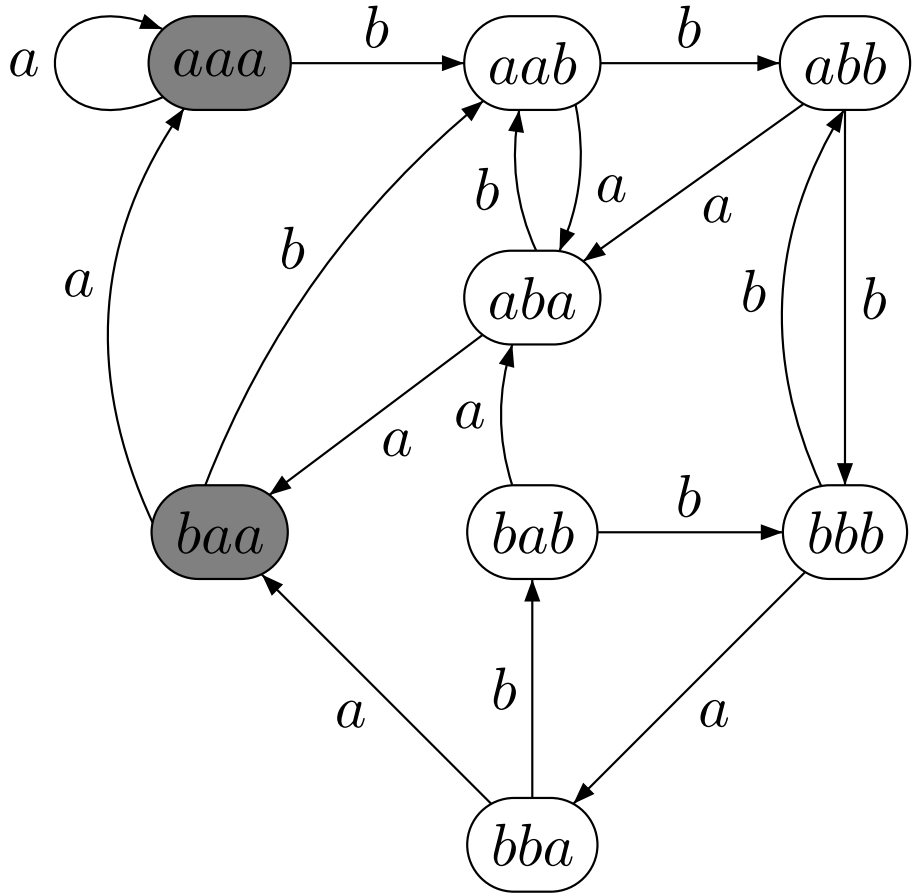
Example

Let $S = \{aa, aba\}$. Then $T = \{aaa, aab, baa, aba\}$.



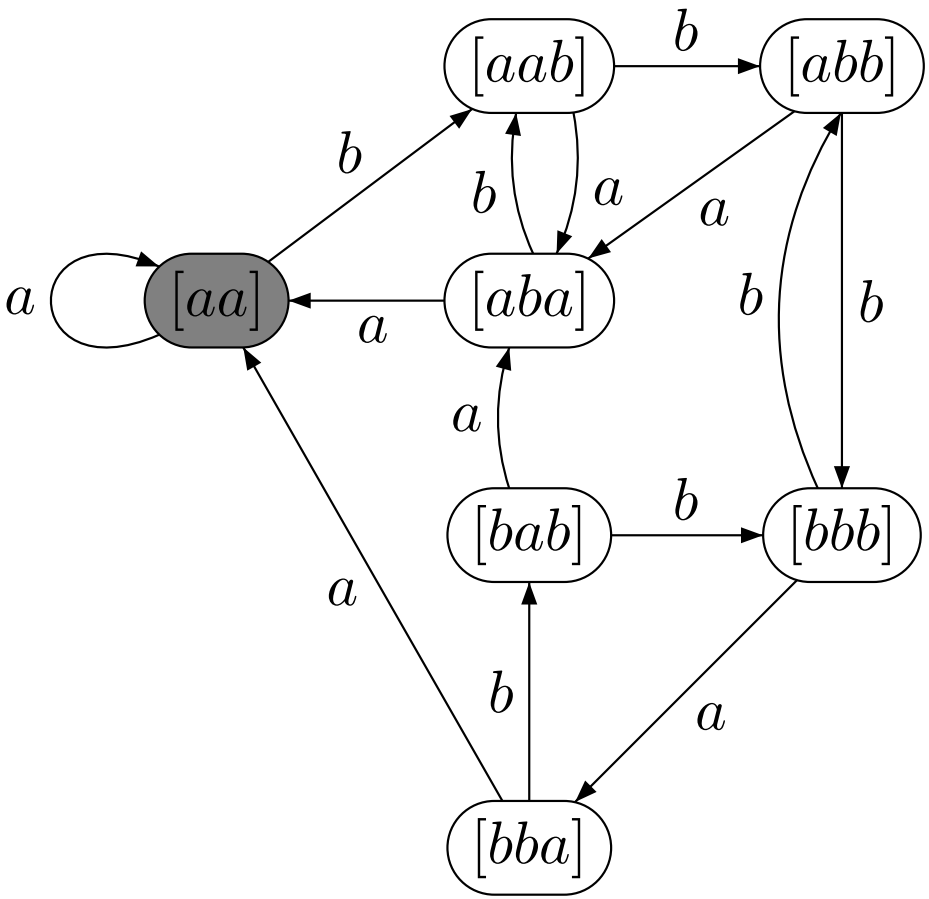
Example

Let $S = \{aa, aba\}$. Then $T = \{aaa, aab, baa, aba\}$.



Example

Let $S = \{aa, aba\}$. Then $T = \{aaa, aab, baa, aba\}$.



Ideal languages generated by two words

Let $\Sigma = \{a, b\}$, and let $u \in \Sigma^n \setminus \{ab^{n-1}, a^{n-1}b, ba^{n-1}, b^{n-1}a\}$, $v \in \Sigma^m \setminus \{ab^{m-1}, a^{m-1}b, ba^{m-1}, b^{m-1}a\}$.

Theorem 4

There is a strongly connected synchronizing automaton $\mathscr{D}_{u,v}$ having $n + m$ states such that $\mathrm{Syn}(\mathscr{D}_{u,v}) = \Sigma^*(u + v)\Sigma^*$.

The construction is based on the minimal automata recognizing languages $\Sigma^* u \Sigma^*$ and $\Sigma^* v \Sigma^*$.

Ideal languages generated by two words

Let $\Sigma = \{a, b\}$, and let $u \in \Sigma^n \setminus \{ab^{n-1}, a^{n-1}b, ba^{n-1}, b^{n-1}a\}$, $v \in \Sigma^m \setminus \{ab^{m-1}, a^{m-1}b, ba^{m-1}, b^{m-1}a\}$.

Theorem 4

There is a strongly connected synchronizing automaton $\mathscr{D}_{u,v}$ having $n + m$ states such that $\mathrm{Syn}(\mathscr{D}_{u,v}) = \Sigma^*(u + v)\Sigma^*$.

The construction is based on the minimal automata recognizing languages $\Sigma^* u \Sigma^*$ and $\Sigma^* v \Sigma^*$.

Ideal languages generated by two words

Let $\Sigma = \{a, b\}$, and let $u \in \Sigma^n \setminus \{ab^{n-1}, a^{n-1}b, ba^{n-1}, b^{n-1}a\}$, $v \in \Sigma^m \setminus \{ab^{m-1}, a^{m-1}b, ba^{m-1}, b^{m-1}a\}$.

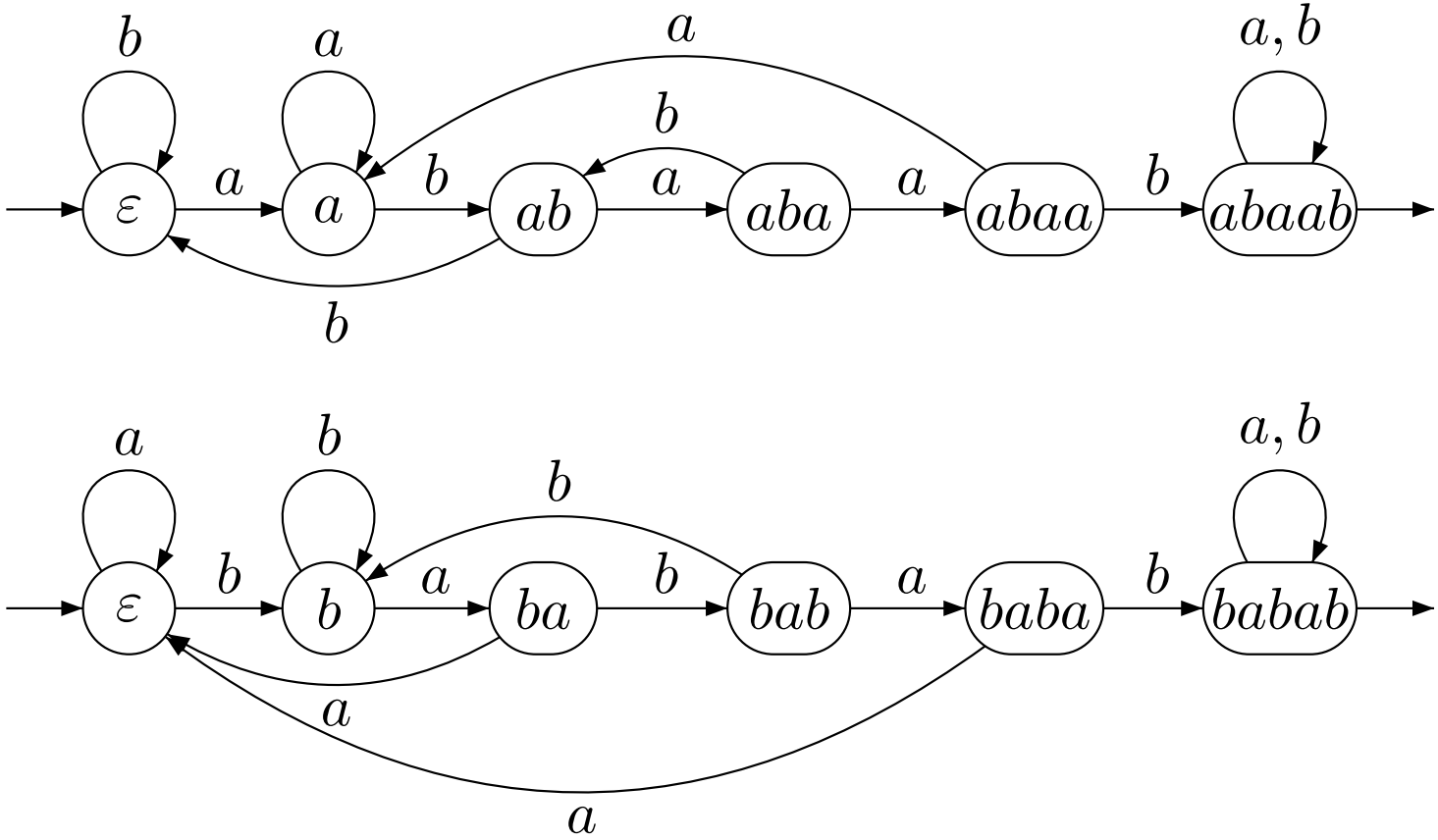
Theorem 4

There is a strongly connected synchronizing automaton $\mathscr{D}_{u,v}$ having $n + m$ states such that $\mathrm{Syn}(\mathscr{D}_{u,v}) = \Sigma^*(u + v)\Sigma^*$.

The construction is based on the minimal automata recognizing languages $\Sigma^* u \Sigma^*$ and $\Sigma^* v \Sigma^*$.

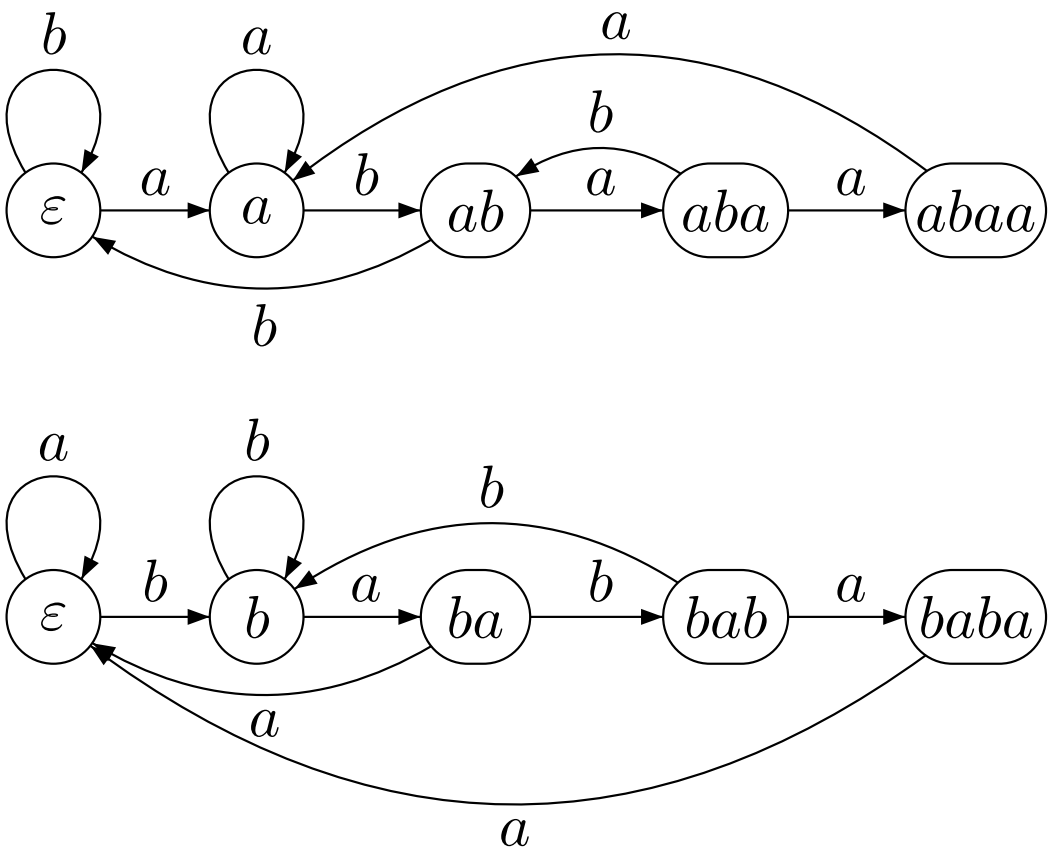
Example

Let $u = abaab$ and $v = babab$.



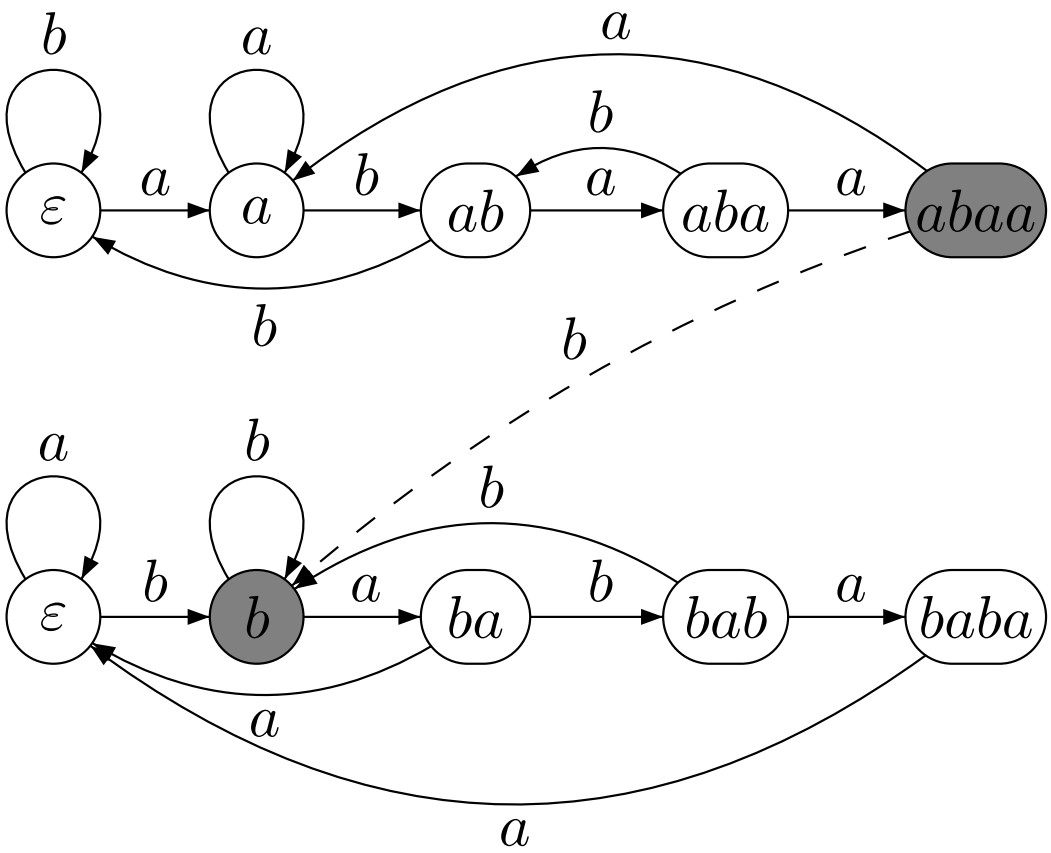
Example

Let $u = abaab$ and $v = babab$.



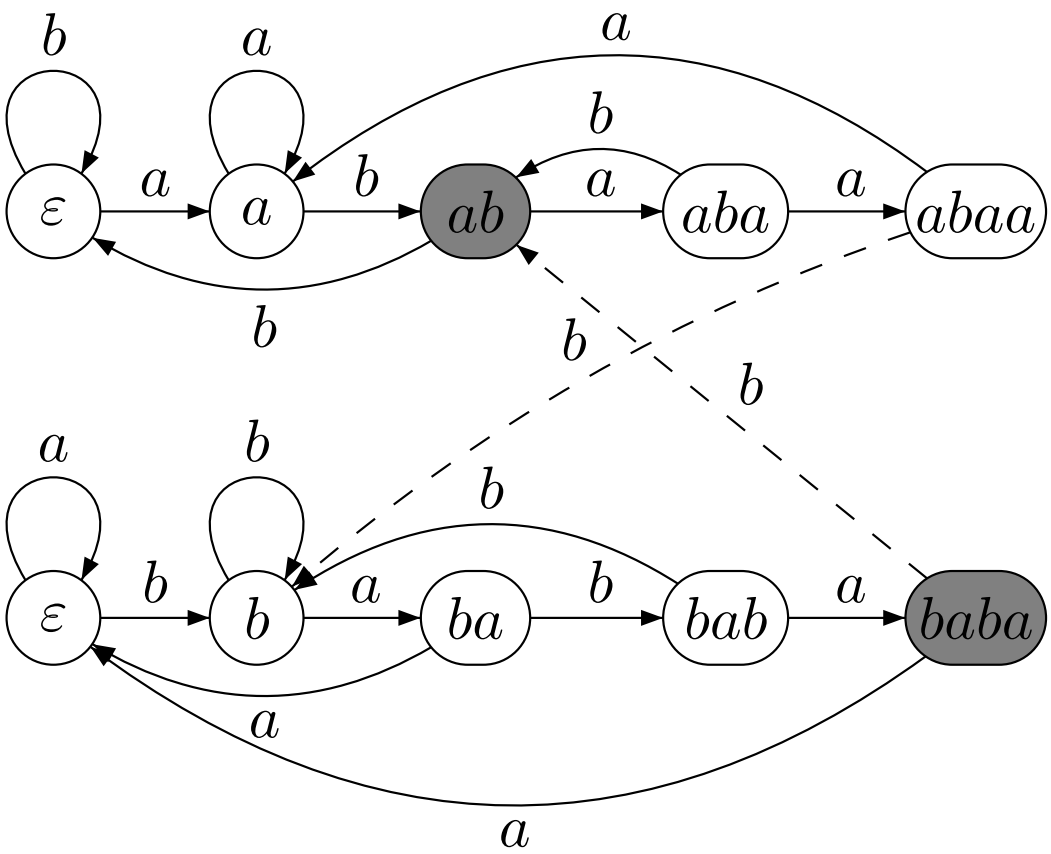
Example

Let $u = abaab$ and $v = babab$.



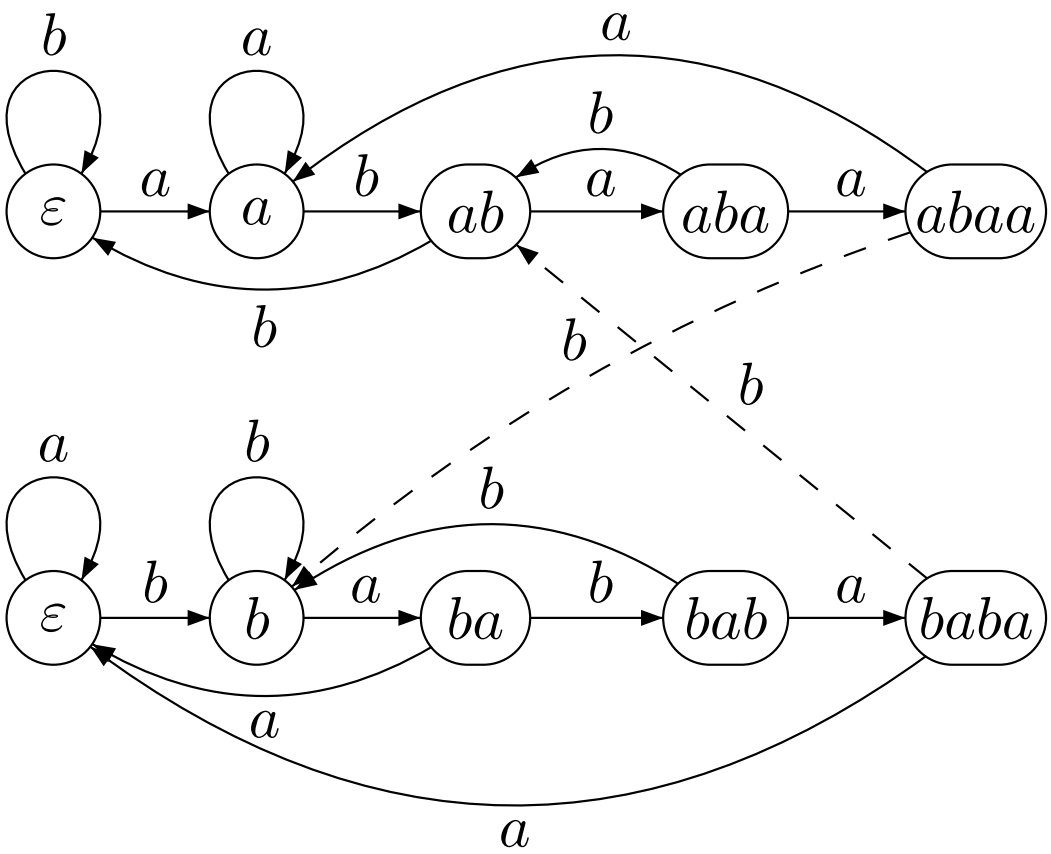
Example

Let $u = abaab$ and $v = babab$.



Example

Let $u = abaab$ and $v = babab$.



Results

- The minimal DFA recognizing an ideal language can be exponentially smaller than a minimal strongly connected synchronizing automaton for which given language serves as the language of reset words.
- An algorithm to construct strongly connected synchronizing automaton with at most 2^n states whose language of reset words is generated by a finite set of words S (n is the maximal length of words in S).

Results

- The minimal DFA recognizing an ideal language can be exponentially smaller than a minimal strongly connected synchronizing automaton for which given language serves as the language of reset words.
- An algorithm to construct strongly connected synchronizing automaton with at most 2^n states whose language of reset words is generated by a finite set of words S (n is the maximal length of words in S).