

Another example. Let us find equivalence classes of R_L for $L = a^*b^*$.

First, what are

$$\text{Ext}(a, L) =$$

$$\text{Ext}(a^{50}, L) =$$

$$\text{Ext}(a^{50}b, L) =$$

$$\text{Ext}(a^{50}b^{20}, L) =$$

$$\text{Ext}(b, L) =$$

$$\text{Ext}(ba, L) =$$

Another example. Let us find equivalence classes of R_L for $L = a^*b^*$.

First, what are

$$\text{Ext}(a, L) = a^*b^*$$

$$\text{Ext}(a^{50}, L) = a^*b^*$$

$$\text{Ext}(a^{50}b, L) = b^*$$

$$\text{Ext}(a^{50}b^{20}, L) = b^*$$

$$\text{Ext}(b, L) = b^*$$

$$\text{Ext}(ba, L) = \emptyset$$

Now, what is $\text{Ext}(w, L)$ if

$$w \in a^* \quad ?$$

$$w \in a^*b^+ \quad ?$$

$$w \notin a^*b^* \quad ?$$

Another example. Let us find equivalence classes of R_L for $L = a^*b^*$.

First, what are

$$\text{Ext}(a, L) = a^*b^*$$

$$\text{Ext}(a^{50}, L) = a^*b^*$$

$$\text{Ext}(a^{50}b, L) = b^*$$

$$\text{Ext}(a^{50}b^{20}, L) = b^*$$

$$\text{Ext}(b, L) = b^*$$

$$\text{Ext}(ba, L) = \emptyset$$

Now, what is $\text{Ext}(w, L)$ if

$$w \in a^* \quad ? \quad a^*b^*$$

$$w \in a^*b^+ \quad ? \quad b^*$$

$$w \notin a^*b^* \quad ? \quad \emptyset$$

So what are the equivalence classes of R_L ?

Another example. Let us find equivalence classes of R_L for $L = a^*b^*$.

First, what are

$$\text{Ext}(a, L) = a^*b^*$$

$$\text{Ext}(a^{50}, L) = a^*b^*$$

$$\text{Ext}(a^{50}b, L) = b^*$$

$$\text{Ext}(a^{50}b^{20}, L) = b^*$$

$$\text{Ext}(b, L) = b^*$$

$$\text{Ext}(ba, L) = \emptyset$$

Now, what is $\text{Ext}(w, L)$ if

$$w \in a^* \quad ? \quad a^*b^*$$

$$w \in a^*b^+ \quad ? \quad b^*$$

$$w \notin a^*b^* \quad ? \quad \emptyset$$

So what are the equivalence classes of R_L ?

$$a^*, a^*b^+, \{u \mid u \text{ is not a prefix of any word in } L\}$$

One more example. Let us find the equivalence classes of R_L for the language

$$L = \{w \in \{a, b\}^* \mid w \text{ contains equally many } a\text{'s and } b\text{'s} \}.$$

One more example. Let us find the equivalence classes of R_L for the language

$$L = \{w \in \{a, b\}^* \mid w \text{ contains equally many } a\text{'s and } b\text{'s} \}.$$

Let us denote

$$|w|_a = \text{number of letters } a \text{ in word } w.$$

Consider languages

$$L_n = \{w \in \{a, b\}^* \mid |w|_a - |w|_b = n\}$$

for different integers n . Clearly $L = L_0$. Every word belongs to exactly one L_n . Observe also that

$$L_n L_m \subseteq L_{n+m}.$$

It follows from the considerations above that if $w \in L_n$ then

$$\text{Ext}(w, L) =$$

One more example. Let us find the equivalence classes of R_L for the language

$$L = \{w \in \{a, b\}^* \mid w \text{ contains equally many } a\text{'s and } b\text{'s}\}.$$

Let us denote

$$|w|_a = \text{number of letters } a \text{ in word } w.$$

Consider languages

$$L_n = \{w \in \{a, b\}^* \mid |w|_a - |w|_b = n\}$$

for different integers n . Clearly $L = L_0$. Every word belongs to exactly one L_n . Observe also that

$$L_n L_m \subseteq L_{n+m}.$$

It follows from the considerations above that if $w \in L_n$ then

$$\text{Ext}(w, L) = L_{-n}$$

Therefore $\text{Ext}(w, L)$ and $\text{Ext}(u, L)$ are identical if and only if w and u belong to the same set L_n . In other words, the equivalence classes of R_L are languages L_n for all $n \in \mathbb{Z}$.

Note that in this case there are **infinitely many equivalence classes**.

Theorem. Language L is regular if and only if the index of R_L is finite. The index is the smallest possible number of states of a DFA recognizing L .

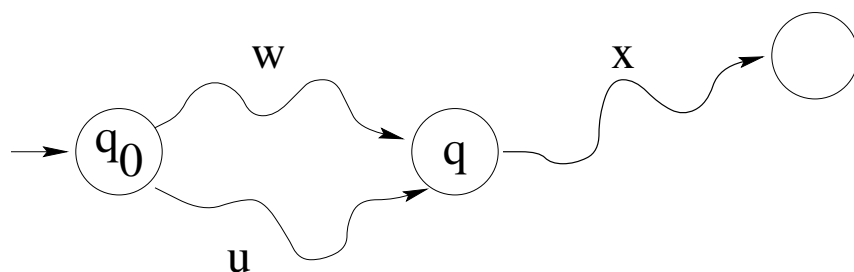
Proof.

Theorem. Language L is regular if and only if the index of R_L is finite. The index is the smallest possible number of states of a DFA recognizing L .

Proof. ($\implies$) Let L be a regular language accepted by DFA A . If w and u are two words that take the automaton to the same state q , i.e.

$$\delta(q_0, w) = \delta(q_0, u),$$

then for every extension x , words wx and ux lead to same state.



This means that wR_Lu .

Since all words leading to same state are in relation R_L , the number of equivalence classes is at most the number of states, and therefore finite.

We have seen that the number of states in a DFA accepting L is at least the index of R_L .

Remark. Assign also to every DFA A its own equivalence relation R_A by

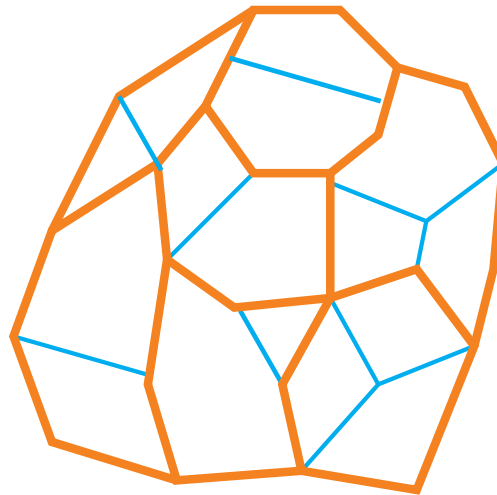
$$w R_A u \iff \delta(q_0, w) = \delta(q_0, u).$$

That is, words w and u are in relation R_A if and only if they take the DFA A to the same state.

We proved that if A recognizes L then the relation R_A is a **refinement** of relation R_L . In other words,

$$w R_A u \implies w R_L u.$$

The equivalence classes of a refinement define a finer partition of the set:



Theorem. Language L is regular if and only if the index of R_L is finite. The index is the smallest possible number of states of a DFA recognizing L .

Proof. ($\Leftarrow$) Assume that R_L has finite index. Let $L_1, L_2, \dots, L_n$ be its equivalence classes. We construct a DFA A with n states that recognizes L . The state set of A is labeled with the equivalence classes $[w]$:

$$Q = \{L_1, L_2, \dots, L_n\}.$$

The idea of the construction is that every input word w will lead to state $[w]$, the equivalence class containing w .

Theorem. Language L is regular if and only if the index of R_L is finite. The index is the smallest possible number of states of a DFA recognizing L .

Proof. ($\Leftarrow$) Assume that R_L has finite index. Let $L_1, L_2, \dots, L_n$ be its equivalence classes. We construct a DFA A with n states that recognizes L . The state set of A is labeled with the equivalence classes $[w]$:

$$Q = \{L_1, L_2, \dots, L_n\}.$$

The idea of the construction is that every input word w will lead to state $[w]$, the equivalence class containing w .

The transitions are defined as follows:

$$\delta([w], a) = [wa].$$

The transitions are well defined, that is, the definition does not depend on which word w we selected from the equivalence class: If $[w] = [u]$ then $[wa] = [ua]$.

(If we would have $[wa] \neq [ua]$ then for some x , $wax \in L$ and $uax \notin L$ or vice versa. But then also $[w] \neq [u]$ since extension ax separates w and u .)

Theorem. Language L is regular if and only if the index of R_L is finite. The index is the smallest possible number of states of a DFA recognizing L .

Proof. ($\Leftarrow$) Assume that R_L has finite index. Let $L_1, L_2, \dots, L_n$ be its equivalence classes. We construct a DFA A with n states that recognizes L . The state set of A is labeled with the equivalence classes $[w]$:

$$Q = \{L_1, L_2, \dots, L_n\}.$$

The idea of the construction is that every input word w will lead to state $[w]$, the equivalence class containing w .

The transitions are defined as follows:

$$\delta([w], a) = [wa].$$

The transitions are well defined, that is, the definition does not depend on which word w we selected from the equivalence class: If $[w] = [u]$ then $[wa] = [ua]$.

Let the initial state q_0 be $[\varepsilon]$. Then

$$\delta(q_0, w) = [w]$$

for every w . The transitions were designed that way:

$$\begin{aligned} \delta([\varepsilon], a_1 a_2 \dots a_n) &= \delta([a_1], a_2 a_3 \dots a_n) = \delta([a_1 a_2], a_3 \dots a_n) \\ &= \dots = [a_1 a_2 \dots a_n]. \end{aligned}$$

Finally, the final states of our automaton are

$$\{[w] \mid w \in L\}.$$

Again, the choice of w does not matter since if $[w] = [u]$ and $w \in L$ then also $u \in L$.

(Otherwise the extension ε would separate them.)

Finally, the final states of our automaton are

$$\{[w] \mid w \in L\}.$$

Again, the choice of w does not matter since if $[w] = [u]$ and $w \in L$ then also $u \in L$.

The automaton we constructed recognizes the language L because

$$\delta(q_0, w) = [w]$$

and $[w]$ is a final state if and only if $w \in L$.

This completes the proof.

Example. Let us construct a DFA for $L = a^*b^*$ based on our knowledge about the equivalence classes of R_L :

$a^*, a^*b^+, \{u \mid u \text{ is not a prefix of any word in } L\}$

Example. Let us construct a DFA for $L = aa + baa$ based on our knowledge about the equivalence classes of R_L :

$\{\varepsilon\}, \{a, ba\}, \{aa, baa\}, \{b\}, \{u \mid u \text{ is not a prefix of any word in } L\}$

Example. Doing the construction on the non-regular language

$$L = \{w \in \{a, b\}^* \mid w \text{ contains equally many } a\text{'s and } b\text{'s} \}$$

leads to infinitely many states, corresponding to the equivalence classes

$$L_n = \{w \in \{a, b\}^* \mid |w|_a - |w|_b = n\}$$

We proved that the smallest possible number of states in a DFA recognizing L is the same as the index of the equivalence relation R_L .

A DFA with this smallest possible number of states is the **minimum-state DFA** of L . This DFA is unique:

Theorem. The minimum-state DFA of a language L is unique (up to renaming states).

Proof.

We proved that the smallest possible number of states in a DFA recognizing L is the same as the index of the equivalence relation R_L .

A DFA with this smallest possible number of states is the **minimum-state DFA** of L . This DFA is unique:

Theorem. The minimum-state DFA of a language L is unique (up to renaming states).

Proof. Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA with n states for the language L , where n is the index of R_L . Let us see how to rename the states so that A becomes identical with the DFA A_0 we constructed in the previous proof.

Recall: the states of A_0 are the equivalence classes $[w]$ of relation R_L and transitions are defined by

$$\delta([w], a) = [wa].$$

As the automaton relation R_A is a refinement of R_L and has no more than n classes, we have that $R_A = R_L = R_{A_0}$. Thus for any words w and u we have

$$[w] = [u] \iff \delta(q_0, w) = \delta(q_0, u)$$

For every $w \in \Sigma^*$ we **rename** the state $\delta(q_0, w)$ as $[w]$.

As the automaton relation R_A is a refinement of R_L and has no more than n classes, we have that $R_A = R_L = R_{A_0}$. Thus for any words w and u we have

$$[w] = [u] \iff \delta(q_0, w) = \delta(q_0, u)$$

For every $w \in \Sigma^*$ we **rename** the state $\delta(q_0, w)$ as $[w]$.

- This renaming is well defined: if $\delta(q_0, w) = \delta(q_0, u)$ then $[w] = [u]$.

(So the name is independent of the choice of the the word w leading to the state.)

As the automaton relation R_A is a refinement of R_L and has no more than n classes, we have that $R_A = R_L = R_{A_0}$. Thus for any words w and u we have

$$[w] = [u] \iff \delta(q_0, w) = \delta(q_0, u)$$

For every $w \in \Sigma^*$ we **rename** the state $\delta(q_0, w)$ as $[w]$.

- This renaming is well defined: if $\delta(q_0, w) = \delta(q_0, u)$ then $[w] = [u]$.
- Every state gets a name, and every equivalence class is a name of a state.

(If a state gets no name then there is no word leading to that state. The state can be removed, yielding a smaller equivalent DFA. This is not possible since n is the smallest possible number of states.)

Every equivalence class $[w]$ is trivially the name of the state $\delta(q_0, w)$.)

As the automaton relation R_A is a refinement of R_L and has no more than n classes, we have that $R_A = R_L = R_{A_0}$. Thus for any words w and u we have

$$[w] = [u] \iff \delta(q_0, w) = \delta(q_0, u)$$

For every $w \in \Sigma^*$ we **rename** the state $\delta(q_0, w)$ as $[w]$.

- This renaming is well defined: if $\delta(q_0, w) = \delta(q_0, u)$ then $[w] = [u]$.
- Every state gets a name, and every equivalence class is a name of a state.
- Different states get different names: if $[w] = [u]$ then $\delta(q_0, w) = \delta(q_0, u)$.

(If p and q get the same name $[w]$ then there is

- a word u such that $\delta(q_0, u) = p$ and $[u] = [w]$, and
- a word v such that $\delta(q_0, v) = q$ and $[v] = [w]$.

But since $[u] = [w] = [v]$ we have that $p = \delta(q_0, u) = \delta(q_0, v) = q$.)

As the automaton relation R_A is a refinement of R_L and has no more than n classes, we have that $R_A = R_L = R_{A_0}$. Thus for any words w and u we have

$$[w] = [u] \iff \delta(q_0, w) = \delta(q_0, u)$$

For every $w \in \Sigma^*$ we **rename** the state $\delta(q_0, w)$ as $[w]$.

- This renaming is well defined: if $\delta(q_0, w) = \delta(q_0, u)$ then $[w] = [u]$.
- Every state gets a name, and every equivalence class is a name of a state.
- Different states get different names: if $[w] = [u]$ then $\delta(q_0, w) = \delta(q_0, u)$.

With this renaming the state set of A becomes the same as the state set of A_0 .

- The **initial state** $q_0 = \delta(q_0, \varepsilon)$ is renamed as $[\varepsilon]$, the initial state of A_0 .
- A state renamed as $[w]$ is a **final state** of A if and only if $w \in L$, which is equivalent to $[w]$ being a final state in A_0 .

As the automaton relation R_A is a refinement of R_L and has no more than n classes, we have that $R_A = R_L = R_{A_0}$. Thus for any words w and u we have

$$[w] = [u] \iff \delta(q_0, w) = \delta(q_0, u)$$

For every $w \in \Sigma^*$ we **rename** the state $\delta(q_0, w)$ as $[w]$.

- This renaming is well defined: if $\delta(q_0, w) = \delta(q_0, u)$ then $[w] = [u]$.
- Every state gets a name, and every equivalence class is a name of a state.
- Different states get different names: if $[w] = [u]$ then $\delta(q_0, w) = \delta(q_0, u)$.

With this renaming the state set of A becomes the same as the state set of A_0 .

- The **initial state** $q_0 = \delta(q_0, \varepsilon)$ is renamed as $[\varepsilon]$, the initial state of A_0 .
- A state renamed as $[w]$ is a **final state** of A if and only if $w \in L$, which is equivalent to $[w]$ being a final state in A_0 .
- With this renaming, **transitions** are exactly as in A_0 : for all $w \in \Sigma^*$, $a \in \Sigma$

$$\delta([w], a) = [wa]$$

(Indeed: this simply states that $\delta(\delta(q_0, w), a) = \delta(q_0, wa)$.)

As the automaton relation R_A is a refinement of R_L and has no more than n classes, we have that $R_A = R_L = R_{A_0}$. Thus for any words w and u we have

$$[w] = [u] \iff \delta(q_0, w) = \delta(q_0, u)$$

For every $w \in \Sigma^*$ we **rename** the state $\delta(q_0, w)$ as $[w]$.

- This renaming is well defined: if $\delta(q_0, w) = \delta(q_0, u)$ then $[w] = [u]$.
- Every state gets a name, and every equivalence class is a name of a state.
- Different states get different names: if $[w] = [u]$ then $\delta(q_0, w) = \delta(q_0, u)$.

With this renaming the state set of A becomes the same as the state set of A_0 .

- The **initial state** $q_0 = \delta(q_0, \varepsilon)$ is renamed as $[\varepsilon]$, the initial state of A_0 .
- A state renamed as $[w]$ is a **final state** of A if and only if $w \in L$, which is equivalent to $[w]$ being a final state in A_0 .
- With this renaming, **transitions** are exactly as in A_0 : for all $w \in \Sigma^*$, $a \in \Sigma$

$$\delta([w], a) = [wa]$$

Conclusion: After the renaming, A and A_0 have the same states, initial and final states and identical transitions. Thus $A = A_0$.

Theorem. There is an algorithm to construct the minimum-state DFA for given regular language L .

Proof 1. A stupid but simple algorithm: enumerate all DFAs in the increasing order of number of states. (First 1-state DFA, then 2-state DFA, etc.) For each DFA A check whether $L(A) = L$. (Equivalence is decidable.) Stop when the first equivalent DFA is found.

Theorem. There is an algorithm to construct the minimum-state DFA for given regular language L .

Proof 2. There is also a practical state minimization algorithm that constructs the minimum-state DFA equivalent to a given DFA A .

1. The first step is to **remove all non-reachable states** from A . State q is non-reachable if there does not exist any input word that takes automaton A to state q .

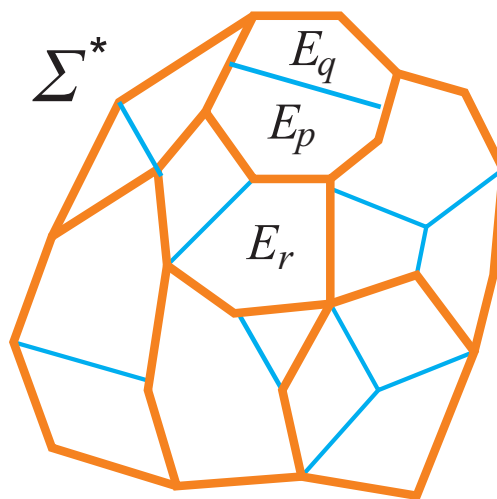
Reachable states can be found by a simple marking procedure: First mark the initial state only. Then mark all states that there is a transition into from some marked state. Continue this until no new states can be marked.

All non-marked states are non-reachable and can be deleted.

2. Now assume the DFA A only contains reachable states. Every state q defines an equivalence class E_q of the automaton relation R_A , consisting of all the words that take A from q_0 to the state q :

$$E_q = \{w \mid \delta(q_0, w) = q\}.$$

We know that R_A is a refinement of the relation R_L :



In the minimum-state DFA the equivalence classes of the automaton are the same as the equivalence classes of R_L . In A we can combine states q and p if their classes E_q and E_p belong to the same equivalence class of R_L . We call such states indistinguishable.

Precisely, p and q are **indistinguishable** if and only if

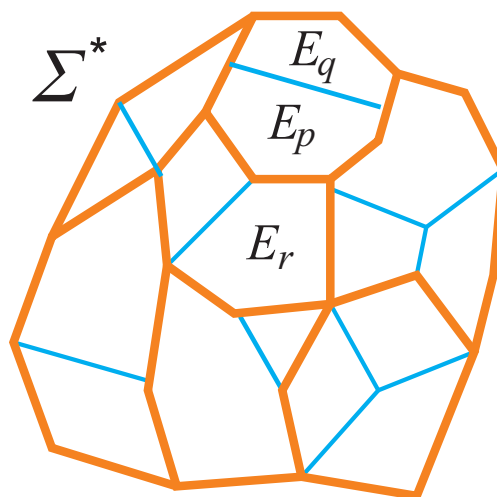
$$(\forall w \in \Sigma^*) \delta(p, w) \in F \iff \delta(q, w) \in F.$$

(It does not make any difference whether the DFA is in state p or q : Exactly the same words lead to a final state. Indistinguishable states can be merged.)

Complementarily, p and q are **distinguishable** if and only if

$$(\exists w \in \Sigma^*) \delta(p, w) \in F \text{ and } \delta(q, w) \notin F \text{ or vice versa.}$$

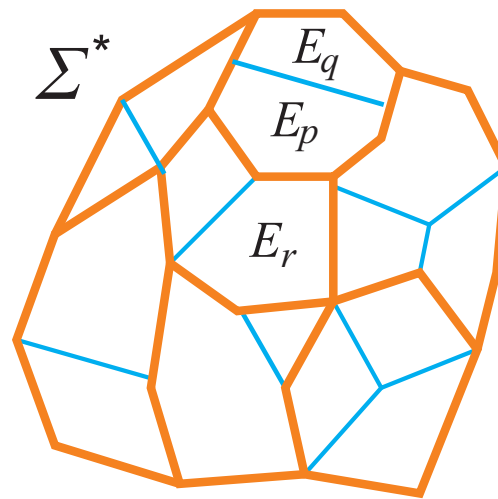
(There is a word w that distinguishes p and q . Distinguishable states cannot be merged.)



Here p and q are indistinguishable, while p and r are distinguishable.

If we find all pairs of indistinguishable states and combine them we end up with a DFA A' whose equivalence relation $R_{A'}$ is identical to R_L . This A' is the minimum-state DFA for language L .

The problem is to find out if two states are indistinguishable. It is easier to find states that are distinguishable. So rather than figuring out which states can be merged we search for states that cannot be merged!



We use the following property:

aw distinguishes p and $q \iff w$ distinguishes $\delta(p, a)$ and $\delta(q, a)$

The algorithm maintains a partition of the state set into classes $C_1, \dots, C_k$ such that for all $i \neq j$ the states of C_i are distinguishable from the states of C_j (even by the same word).

We use the following property:

$$aw \text{ distinguishes } p \text{ and } q \iff w \text{ distinguishes } \delta(p, a) \text{ and } \delta(q, a)$$

The algorithm maintains a partition of the state set into classes $C_1, \dots, C_k$ such that for all $i \neq j$ the states of C_i are distinguishable from the states of C_j (even by the same word).

- In the beginning we start with two classes: F (the set of final states) and $Q \setminus F$ (the set of non-final states). The empty word ε distinguishes these classes.

We use the following property:

$$aw \text{ distinguishes } p \text{ and } q \iff w \text{ distinguishes } \delta(p, a) \text{ and } \delta(q, a)$$

The algorithm maintains a partition of the state set into classes $C_1, \dots, C_k$ such that for all $i \neq j$ the states of C_i are distinguishable from the states of C_j (even by the same word).

- In the beginning we start with two classes: F (the set of final states) and $Q \setminus F$ (the set of non-final states). The empty word ε distinguishes these classes.
- At each round we look for a class C and a letter a such that for some $p, q \in C$ the states $\delta(q, a)$ and $\delta(p, a)$ belong to different classes. In this case, because $\delta(q, a)$ and $\delta(p, a)$ are distinguished by some word w , we can distinguish q and p by word aw . Thus we split C in such a way that any $p, q \in C$ go in the same part if and only if $\delta(q, a)$ and $\delta(p, a)$ are in the same class.

The algorithm is repeated until no class can be split further.

Claim: At the end all states that are in the same class are indistinguishable.

Proof. Assume the contrary: let w be the shortest word that distinguishes some states (say p and q) that belong to the same class (say C). Clearly $w \neq \varepsilon$ because otherwise p would be final and q non-final, or vice versa, but such pairs were put in different classes in the beginning.

So $w = au$ for some letter a .

But then $\delta(q, a)$ and $\delta(p, a)$ are distinguished by u that is shorter than w , and therefore $\delta(q, a)$ and $\delta(p, a)$ are in different classes. But this means that the class C can be split, contradicting the assumption that no classes can be split further.