

Closure properties

It follows from the definition of regular expressions that the **union** $L_1 \cup L_2$ of any two regular languages L_1 and L_2 is regular. Namely, if r_1 and r_2 are regular expressions for L_1 and L_2 then $r_1 + r_2$ is a regular expression for the union $L_1 \cup L_2$.

In the same way, the **concatenation** of two regular languages is always regular: $r_1 r_2$ is a regular expression for the concatenation.

We say that the family of regular languages is **closed** under union and concatenation.

In general, let Op denote some **language operation**, that is, an operation whose operands and result are all formal languages.

We say that a family $\mathcal{F}$ of languages is **closed under the operation Op** if

$$L_1, L_2, \dots \in \mathcal{F} \implies Op(L_1, L_2, \dots) \in \mathcal{F}.$$

(*E.g.*, the family of regular languages is closed under operation Op if the result after applying Op on regular languages is always a regular language.)

In general, let Op denote some **language operation**, that is, an operation whose operands and result are all formal languages.

We say that a family $\mathcal{F}$ of languages is **closed under the operation Op** if

$$L_1, L_2, \dots \in \mathcal{F} \implies Op(L_1, L_2, \dots) \in \mathcal{F}.$$

(*E.g.*, the family of regular languages is closed under operation Op if the result after applying Op on regular languages is always a regular language.)

Next we investigate the closure properties of regular languages under some common operations. We start the familiar boolean operations: **union**, **intersection** and **complement**. We already know the closure under union.

Consider next the **complement**. Let $L \subseteq \Sigma^*$ be a regular language over alphabet Σ . Is its complement

$$\bar{L} = \Sigma^* \setminus L = \{w \in \Sigma^* \mid w \notin L\}$$

necessarily regular?

Consider next the **complement**. Let $L \subseteq \Sigma^*$ be a regular language over alphabet Σ . Is its complement

$$\bar{L} = \Sigma^* \setminus L = \{w \in \Sigma^* \mid w \notin L\}$$

necessarily regular?

The answer is **yes**, the family of regular languages is closed under complement.

Consider a **DFA A that recognizes L** . Without loss of generality we may assume that A uses alphabet Σ , the same alphabet relative to which the complement is defined.

Every input word w takes A to some state q . The word w is accepted if and only if state q is among the final states.

Let us change A by making every final state a non-final state, and vice versa:

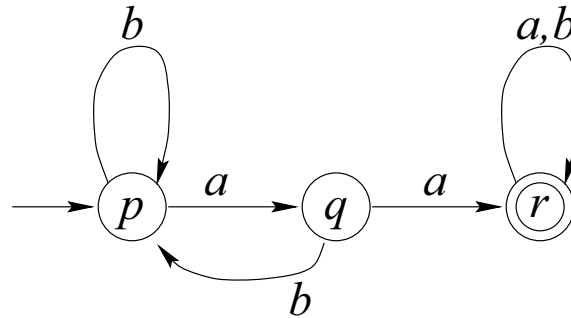
final $\longleftrightarrow$ non-final

The new DFA accepts word w if and only if the original DFA did not accept w . So the new DFA recognizes the complement of L .

Example. Let us find a DFA for the complement

$$\{a, b\}^* \setminus L(A)$$

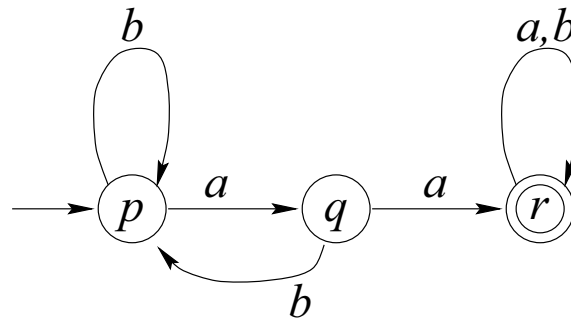
of the language $L(A)$ recognized by DFA



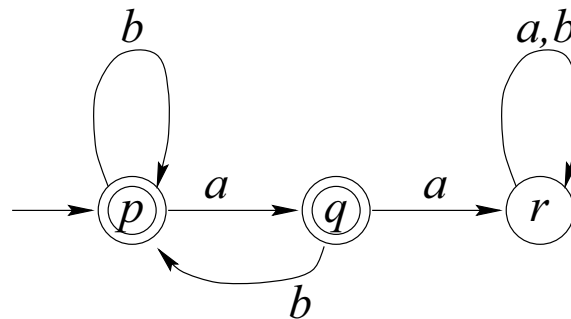
Example. Let us find a DFA for the complement

$$\{a, b\}^* \setminus L(A)$$

of the language $L(A)$ recognized by DFA



This DFA recognizes the complement of $L(A)$:



What about **intersection** ? Elementary set theory (**de Morgan's law**) tells us how to use union and complement to do intersection. For any languages L_1 and L_2

$$L_1 \cap L_2 = \overline{\bar{L}_1 \cup \bar{L}_2}$$

So

$$L_1 \text{ and } L_2 \text{ regular} \implies \bar{L}_1 \text{ and } \bar{L}_2 \text{ regular} \implies \bar{L}_1 \cup \bar{L}_2 \text{ regular} \implies \overline{\bar{L}_1 \cup \bar{L}_2} \text{ regular}$$

Remark. All closures so far are **effective** in the sense that we can algorithmically construct the result of the operation on any given regular languages.

- The union can be constructed by inserting a plus sign between the regular expressions of the input languages,
- the complement can be formed by swapping the final states of a DFA,
- the intersection can be done by performing a sequence of effective operations for union and complementation.

It is irrelevant which description we use for the operands: they may be given as a DFA, NFA, ε -NFA or regular expression. All formats are effectively equivalent since we can mechanically convert any device into an equivalent device of any other type.

We say that the family of regular languages is **effectively closed** under operation Op if there exists an algorithm (=mechanical procedure) that produces the result of the operation for any given regular input languages. The format of the inputs and outputs can be arbitrary since we have algorithms for converting from one format to any other.

We have proved the following theorem:

Theorem. The family of regular languages is effectively closed under union, concatenation, Kleene star, complementation and intersection.

The closure properties can be used in proving languages regular. It is enough to show how to build the language from known regular languages using above operations.

Example. The language L containing all words over the English alphabet

$$\Sigma = \{a, b, c, \dots, z\}$$

that do not contain the word utu as a subword is regular. Namely, it is the complement of the language

$$\Sigma^*utu\Sigma^*$$

of all words that do contain utu as a subword.

We can also use closure properties to show that some languages are **not** regular.

Example. Let us show that

$$L = \{a^n c^m b^n \mid n, m \geq 0\}$$

is not regular.

Assume the contrary: assume that L is regular. Then also the language

$$L \cap a^* b^*$$

is regular because the family of regular languages is closed under intersection and $a^* b^*$ is regular. But we already know that

$$L \cap a^* b^* = \{a^n b^n \mid n \geq 0\}$$

is **not** regular. Therefore our assumption that L is regular has to be false.

Another example. The language

$$L = \{a^{p-1} \mid p \text{ is a prime number} \}$$

is not regular.

+ If you start with known regular languages, apply operations (under which reg. languages are closed) and end up with language L , then L is a regular language.

+ If you start with L , apply operations to L and known regular languages, and end up with a language that is known to be non-regular, then L is non-regular.

+ If you start with known regular languages, apply operations (under which reg. languages are closed) and end up with language L , then L is a regular language.

+ If you start with L , apply operations to L and known regular languages, and end up with a language that is known to be non-regular, then L is non-regular.

BUT DO NOT USE THE WRONG DIRECTION:

– Starting with L and applying operations on L and regular languages you end up with a known regular language. This does not prove a thing about L : it can be either regular or non-regular.

– Starting with known non-regular languages you apply operations and end up with L . Also this does not prove anything about L .

Next we introduce some new language operations.

Let Σ and Δ be two alphabets. A **homomorphism** is a function

$$h : \Sigma \longrightarrow \Delta^*,$$

that assigns to each letter $a \in \Sigma$ a word $h(a) \in \Delta^*$.

Homomorphisms are **applied to words** by coding each letter separately and concatenating the results together:

$$h(a_1 a_2 \dots a_n) = h(a_1) h(a_2) \dots h(a_n)$$

Example. Homomorphism

$$h(a) = 0,$$

$$h(b) = 01$$

is from $\Sigma = \{a, b\}$ to $\Delta = \{0, 1\}$. We have

$$h(ba) =$$

$$h(babba) =$$

$$h(\varepsilon) =$$

Example. Homomorphism

$$h(a) = 0,$$

$$h(b) = 01$$

is from $\Sigma = \{a, b\}$ to $\Delta = \{0, 1\}$. We have

$$h(ba) = 010$$

$$h(babba) = 01001010$$

$$h(\varepsilon) = \varepsilon$$

Homomorphisms are **applied to languages** as well: The homomorphic image $h(L)$ of a language L consists of all homomorphic images of all L 's words:

$$h(L) = \{h(w) \mid w \in L\} = \bigcup_{w \in L} \{h(w)\}$$

Homomorphisms are **applied to languages** as well: The homomorphic image $h(L)$ of a language L consists of all homomorphic images of all L 's words:

$$h(L) = \{h(w) \mid w \in L\} = \bigcup_{w \in L} \{h(w)\}$$

Example. Our sample homomorphism

$$h(a) = 0,$$

$$h(b) = 01$$

gives

$$h(b + ab + bbb) = h(\{b, ab, bbb\}) =$$

$$h(b^*) = h(\{\varepsilon, b, bb, bbb, \dots\}) =$$

$$h(a^*b + bb) =$$

Homomorphisms are **applied to languages** as well: The homomorphic image $h(L)$ of a language L consists of all homomorphic images of all L 's words:

$$h(L) = \{h(w) \mid w \in L\} = \bigcup_{w \in L} \{h(w)\}$$

Example. Our sample homomorphism

$$h(a) = 0,$$

$$h(b) = 01$$

gives

$$h(b + ab + bbb) = h(\{b, ab, bbb\}) = \{01, 001, 010101\} = 01 + 001 + 010101$$

$$h(b^*) = h(\{\varepsilon, b, bb, bbb, \dots\}) = \{\varepsilon, 01, 0101, 010101, \dots\} = (01)^*$$

$$h(a^*b + bb) = 0^*01 + 0101$$

A **substitution** is a generalization of the notion of homomorphism. A substitution assigns a **language to each letter**. For example

$$s(a) = 0 + 11$$

$$s(b) = 0^*10^*$$

A **substitution** is a generalization of the notion of homomorphism. A substitution assigns a **language to each letter**. For example

$$\begin{aligned}s(a) &= 0 + 11 \\ s(b) &= 0^*10^*\end{aligned}$$

A substitution s is **regular** if $s(a)$ is a regular language for each $a \in \Sigma$. It is a **finite** substitution if $s(a)$ is a finite language for each $a \in \Sigma$.

A **substitution** is a generalization of the notion of homomorphism. A substitution assigns a **language to each letter**. For example

$$\begin{aligned}s(a) &= 0 + 11 \\ s(b) &= 0^*10^*\end{aligned}$$

A substitution s is **regular** if $s(a)$ is a regular language for each $a \in \Sigma$. It is a **finite** substitution if $s(a)$ is a finite language for each $a \in \Sigma$.

Substitution s is **applied to words** by applying it to each letter separately and concatenating the results (which are languages!) together:

$$s(a_1a_2 \dots a_n) = s(a_1)s(a_2) \dots s(a_n).$$

A **substitution** is a generalization of the notion of homomorphism. A substitution assigns a **language to each letter**. For example

$$\begin{aligned}s(a) &= 0 + 11 \\ s(b) &= 0^*10^*\end{aligned}$$

A substitution s is **regular** if $s(a)$ is a regular language for each $a \in \Sigma$. It is a **finite** substitution if $s(a)$ is a finite language for each $a \in \Sigma$.

Substitution s is **applied to words** by applying it to each letter separately and concatenating the results (which are languages!) together:

$$s(a_1a_2 \dots a_n) = s(a_1)s(a_2) \dots s(a_n).$$

Finally, substitution s **applied to a language** L is the language consisting of all words that can be obtained by applying the substitution on L 's words:

$$s(L) = \{u \mid u \in s(w) \text{ for some } w \in L\} = \bigcup_{w \in L} s(w).$$

A **substitution** is a generalization of the notion of homomorphism. A substitution assigns a **language to each letter**. For example

$$\begin{aligned}s(a) &= 0 + 11 \\ s(b) &= 0^*10^*\end{aligned}$$

A substitution s is **regular** if $s(a)$ is a regular language for each $a \in \Sigma$. It is a **finite** substitution if $s(a)$ is a finite language for each $a \in \Sigma$.

Substitution s is **applied to words** by applying it to each letter separately and concatenating the results (which are languages!) together:

$$s(a_1a_2 \dots a_n) = s(a_1)s(a_2) \dots s(a_n).$$

Finally, substitution s **applied to a language** L is the language consisting of all words that can be obtained by applying the substitution on L 's words:

$$s(L) = \{u \mid u \in s(w) \text{ for some } w \in L\} = \bigcup_{w \in L} s(w).$$

Remark: Every homomorphism is a special type of substitution where each $s(a)$ consists of a single word.

Example.

$$s(a) = 0 + 11$$

$$s(b) = 0^*10^*$$

$$f(a) = 000 + 11$$

$$f(b) = 01 + \varepsilon$$

are substitutions from $\Sigma = \{a, b\}$ to $\Delta = \{0, 1\}$.

Here, s is a regular substitution and f is a finite substitution. (Of course, every finite substitution is also regular.)

Example.

$$s(a) = 0 + 11$$

$$s(b) = 0^*10^*$$

$$f(a) = 000 + 11$$

$$f(b) = 01 + \varepsilon$$

are substitutions from $\Sigma = \{a, b\}$ to $\Delta = \{0, 1\}$.

Here, s is a regular substitution and f is a finite substitution. (Of course, every finite substitution is also regular.)

$$f(ba) =$$

$$s(aba) =$$

$$s(\varepsilon) =$$

Example.

$$s(a) = 0 + 11$$

$$s(b) = 0^*10^*$$

$$f(a) = 000 + 11$$

$$f(b) = 01 + \varepsilon$$

are substitutions from $\Sigma = \{a, b\}$ to $\Delta = \{0, 1\}$.

Here, s is a regular substitution and f is a finite substitution. (Of course, every finite substitution is also regular.)

$$f(ba) = (01 + \varepsilon)(000 + 11) = \{01000, 0111, 000, 11\}$$

$$s(aba) = (0 + 11)0^*10^*(0 + 11)$$

$$s(\varepsilon) = \{\varepsilon\}$$

$$s(ba + aba) = s(ba) \cup s(aba) =$$

$$f(ba^*) =$$

$$s(ab + b^*) =$$

$$s(\varepsilon) =$$

$$s(\emptyset) =$$

Example.

$$s(a) = 0 + 11$$

$$s(b) = 0^*10^*$$

$$f(a) = 000 + 11$$

$$f(b) = 01 + \varepsilon$$

are substitutions from $\Sigma = \{a, b\}$ to $\Delta = \{0, 1\}$.

Here, s is a regular substitution and f is a finite substitution. (Of course, every finite substitution is also regular.)

$$f(ba) = (01 + \varepsilon)(000 + 11) = \{01000, 0111, 000, 11\}$$

$$s(aba) = (0 + 11)0^*10^*(0 + 11)$$

$$s(\varepsilon) = \{\varepsilon\}$$

$$s(ba + aba) = s(ba) \cup s(aba) = 0^*10^*(0 + 11) + (0 + 11)0^*10^*(0 + 11)$$

$$f(ba^*) = (01 + \varepsilon)(000 + 11)^*$$

$$s(ab + b^*) = (0 + 11)0^*10^* + (0^*10^*)^*$$

$$s(\varepsilon) = \{\varepsilon\}$$

$$s(\emptyset) = \emptyset$$

Lemma. For any substitution s and languages L and K

$$s(L \cup K) = s(L) \cup s(K),$$

$$s(LK) = s(L)s(K),$$

$$s(L^*) = s(L)^*.$$

Lemma. For any substitution s and languages L and K

$$\begin{aligned}s(L \cup K) &= s(L) \cup s(K), \\ s(LK) &= s(L)s(K), \\ s(L^*) &= s(L)^*.\end{aligned}$$

Corollary. If L is a regular language and s a regular substitution then $s(L)$ is (effectively) a regular language. In other words, the family of regular languages is effectively closed under regular substitutions.

Proof. Induction on the size of the regular expression r for the language L .

Lemma. For any substitution s and languages L and K

$$\begin{aligned}s(L \cup K) &= s(L) \cup s(K), \\ s(LK) &= s(L)s(K), \\ s(L^*) &= s(L)^*.\end{aligned}$$

Corollary. If L is a regular language and s a regular substitution then $s(L)$ is (effectively) a regular language. In other words, the family of regular languages is effectively closed under regular substitutions.

Proof. Induction on the size of the regular expression r for the language L .

A regular expression for $s(L)$ is obtained by replacing every occurrence of every letter a in the regular expression r for L by the regular expression r_a for the language $s(a)$. For example,

$$\begin{aligned}s(a^*b + bb) &= s(a^*b) + s(bb) \\ &= s(a^*)s(b) + s(b)s(b) \\ &= s(a)^*s(b) + s(b)s(b) \\ &= r_a^*r_b + r_br_b.\end{aligned}$$

Corollary. The family of regular languages is closed under homomorphisms.

Proof. Every homomorphism is a regular substitution.

Corollary. The family of regular languages is closed under homomorphisms.

Proof. Every homomorphism is a regular substitution.

Example. Using homomorphisms and substitutions

$$h(a) = 0$$

$$h(b) = 01$$

$$s(a) = 0 + 11$$

$$s(b) = 0^*10^*$$

$$f(a) = 000 + 11$$

$$f(b) = 01 + \varepsilon$$

we have

$$h((ab^*aa + b)^* + aba) =$$

$$s(ab^* + bba) =$$

$$f(\emptyset + \varepsilon + a) =$$

Corollary. The family of regular languages is closed under homomorphisms.

Proof. Every homomorphism is a regular substitution.

Example. Using homomorphisms and substitutions

$$h(a) = 0$$

$$h(b) = 01$$

$$s(a) = 0 + 11$$

$$s(b) = 0^*10^*$$

$$f(a) = 000 + 11$$

$$f(b) = 01 + \varepsilon$$

we have

$$h((ab^*aa + b)^* + aba) = (0(01)^*00 + 01)^* + 0010$$

$$s(ab^* + bba) = (0 + 11)(0^*10^*)^* + 0^*10^*0^*10^*(0 + 11)$$

$$f(\emptyset + \varepsilon + a) = \emptyset + \varepsilon + 000 + 11$$

Closure under homomorphisms and regular substitutions can be used in proving non-regularity of languages.

Example. Let us prove that

$$L = \{(ab)^p \mid p \text{ is a prime number} \}$$

is not regular. We already know that

$$L_p = \{a^p \mid p \text{ is a prime number} \}$$

is not regular, so let us **reduce** L into L_p .

Closure under homomorphisms and regular substitutions can be used in proving non-regularity of languages.

Example. Let us prove that

$$L = \{(ab)^p \mid p \text{ is a prime number} \}$$

is not regular. We already know that

$$L_p = \{a^p \mid p \text{ is a prime number} \}$$

is not regular, so let us **reduce** L into L_p .

Assume that L is regular. Then also $h(L)$ is regular where h is the homomorphism

$$\begin{aligned} h(a) &= a, \\ h(b) &= \varepsilon. \end{aligned}$$

But $h(L) = L_p$, which is not regular. Therefore L is not regular either.

Our next language operation: the **inverse homomorphism**.

Let h be a homomorphism from Σ to Δ . The inverse homomorphism h^{-1} is defined on languages of alphabet Δ .

For any $L \subseteq \Delta^*$,

$$h^{-1}(L) = \{w \in \Sigma^* \mid h(w) \in L\}$$

(The language $h^{-1}(L)$ consists of all words over Σ that are mapped by the homomorphism h into L .)

If L contains only one word we may simply write $h^{-1}(w)$ instead of $h^{-1}(\{w\})$.

Example. Using the homomorphism

$$\begin{aligned}g(a) &= 01, \\g(b) &= 011, \\g(c) &= 101\end{aligned}$$

we have

$$\begin{aligned}g^{-1}(011101) &= \\g^{-1}(01101) &= \\g^{-1}(010) &= \\g^{-1}((10)^*1) &= \end{aligned}$$

Example. Using the homomorphism

$$\begin{aligned}g(a) &= 01, \\g(b) &= 011, \\g(c) &= 101\end{aligned}$$

we have

$$\begin{aligned}g^{-1}(011101) &= \{bc\} \\g^{-1}(01101) &= \{ac, ba\} \\g^{-1}(010) &= \emptyset \\g^{-1}((10)^*1) &= ca^*\end{aligned}$$

Theorem. The family of regular languages is (effectively) closed under inverse homomorphism.

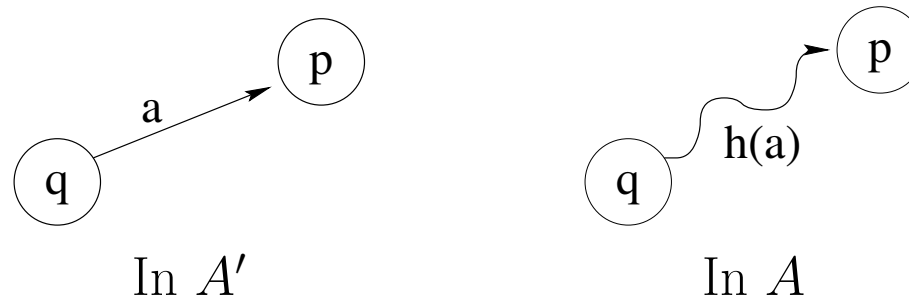
Proof. Let L be a regular language, and h a homomorphism. Let A be a DFA recognizing L . We construct a DFA A' that recognizes $h^{-1}(L)$.

Theorem. The family of regular languages is (effectively) closed under inverse homomorphism.

Proof. Let L be a regular language, and h a homomorphism. Let A be a DFA recognizing L . We construct a DFA A' that recognizes $h^{-1}(L)$.

A' has the same states as A . Also the initial state and final states are identical. Only the transitions are different: an input symbol a from state q takes A' into the same state that the input word $h(a)$ takes A from the same state q :

$$\delta'(q, a) = \delta(q, h(a)).$$



Computations by A' simulate computations by A : After reading input

$$w = a_1 a_2 \dots a_n$$

A' is in the same state as A is after reading the input

$$h(w) = h(a_1)h(a_2) \dots h(a_n).$$

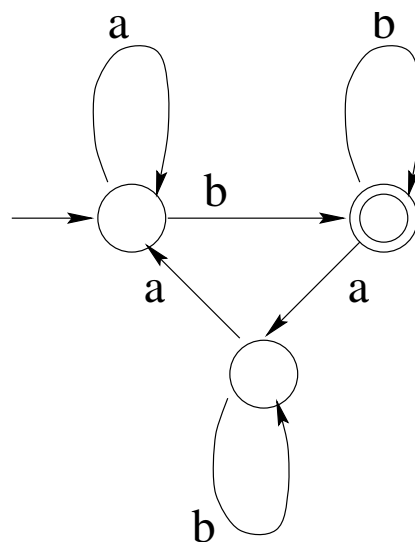
Since the same states are final states in both machines, DFA A' accepts w if and only if DFA A accepts $h(w)$.

In other words, the language recognized by A' is $h^{-1}(L)$.

Example. Let

$$\begin{aligned}h(0) &= ab, \\h(1) &= abb, \\h(2) &= bab,\end{aligned}$$

and let language L be defined by the DFA A



Let us construct a DFA A' that recognizes $h^{-1}(L)$.