Closure Properties of Regular Languages

**Theorem.** The class of regular languages is closed under union, intersection, subtraction, complementation, concatenation, Kleene closure and reversal.

**Note:** closure under union, concatenation and Kleene closure comes from the fact that regular languages are represented by regular expressions; closure under intersection, subtraction and complementation comes from the fact that regular languages are represented by DFAs. The only part still missing is the closure under reversal.
Closure Properties of Regular Languages

**Proof.** We could prove it by structural induction (on the number of operators applied), since $\emptyset^R = \emptyset$, $\{\varepsilon\}^R = \{\varepsilon\}$ and $\{a\}^R = \{a\}$ for every $a \in \Sigma$, and the extension through union, concatenation and Kleene closure is not hard.

We will prove it by construction, instead:

- Assume $M$ is an NFA (or DFA), with language $L(M)$.
- We construct and NFA $M'$ such that $L(M') = L(M)^R$.
- The equivalence of DFAs and NFAs will do the rest. This is the only proof (of any of the statements of the theorem) where NFAs are really used.
Details: Let \( M = (Q, \Sigma, \delta, q_0, F) \). We construct \( M' \) as follows: if \(|F| > 1\), introduce a new state \( s \) and connect every element in \( F \) to it via an \( \epsilon \)-transition. Otherwise, let \( s \) be the single state in \( F \).

Define: \( M' = (Q \cup \{s\}, \Sigma, \delta', \{s\}, \{q_0\}) \), where \( q' \in \delta'(q, a) \iff q \in \delta(q', a), \forall a \in \{\epsilon\} \cup \Sigma \).

All we have done is to "reverse the arrows", after a small adjustment. We have an NFA (we cannot expect a DFA, even if \( M \) was a DFA with a singleton accepting state - why?) whose accepting paths are exactly the reversal of the accepting paths of the original.
Closure Properties of Regular Languages - Substitutions

Substitution. Let \( f \) be a mapping: \( f(a) = L_a \), where \( a \in \Sigma \) and \( L_a \) is a language over an alphabet \( \Gamma \). Extend the function \( f \) to \( \Sigma^* \) by \( f(\varepsilon) = \{ \varepsilon \} \) and \( f(a_1a_2\ldots a_k) = f(a_1)f(a_2)\ldots f(a_k) \) for \( a_1, \ldots, a_k \in \Sigma \). For any language \( L \subseteq \Sigma^* \), we apply \( f \) to \( L \): \( f(L) = \bigcup_{x \in L} f(x) \).

**Ex.**: \( L = \{01, 10\} \); \( f(0) = 0(0+1)^* \); \( f(1) = (0+1)^*1 \).

\[
\begin{align*}
f(L) &= f(01) \cup f(10) = f(0)f(1) \cup f(1)f(0) \\
      &= 0(0+1)^*(0+1)^*1 \cup (0+1)^*10(0+1)^* \\
      &= 0(0+1)^*1 \cup (0+1)^*10(0+1)^*
\end{align*}
\]

A substitution is called a homomorphism if, for any \( a \in \Sigma \), \( f(a) \) is a language with a single string (= a singleton string).
Closure Properties of Regular Languages - Substitutions

**Proposition.** Let $f$ be a substitution over $\Sigma$; assume that $L \in \Sigma^*$ is a regular language and that $f(a)$ is a regular language for each $a \in \Sigma$. Then $f(L)$ is a regular language.

**Proof.** Let $r$ be a RE for $L$, $r_a$ a RE for $f(a)$ for each $a \in \Sigma$. Replace each occurrence of $a$ in $r$ by $f(a)$. We obtain a new regular expression $r'$.

We observe:

a) For any two sets $A, B \subseteq \Sigma^*$, $f(A \cup B) = f(A) \cup f(B)$, $f(A \cdot B) = f(A) \cdot f(B)$, and $f(A^*) = f(A)^*$.

b) For any two regular expressions $r$ and $s$, $(r + s)' = r' + s'$, $(rs)' = r's'$, and $(r^*)' = (r')^*$.

This could use detailed proof as set inclusions and RE equalities…
Finite Automata

Closure Properties of Regular Languages - Substitutions

We now start the induction:

The base cases (0 operators):

- $f(\emptyset) = \emptyset$ (extending the definition to the empty language). Regularity is preserved: the empty language is regular and its image (also the empty language) is also regular.

- $f(\{\epsilon\}) = \{\epsilon\}$. The image of the language consisting of the empty string is a regular language.

- $f(\{a\})$ is a regular language for each $a \in \Sigma$, by definition of $f$.

All three basis steps give rise to regular languages and their regular expressions starting from the regular language building blocks and their regular expressions.
Finite Automata

Closure Properties of Regular Languages

The structural induction step: Let $r$ and $s$ be regular expressions defining regular languages $A$ and $B \subseteq \Sigma^*$ such that $f(A)$ and $f(B)$ are regular languages with regular expressions $r'$ and $s'$. Assume the result holds for sets constructed with up to $n$ operators; prove it holds if you apply any one of the operators to two such sets.

We have:

$$r + s \rightarrow A \cup B : f(A \cup B) = f(A) \cup f(B) \rightarrow r' + s' = (r+s)'$$

and the image language is regular. Identical arguments will work for the remaining structural operations (concatenation and Kleene closure).
Closure Properties of Regular Languages - Quotients

Definition. The quotient of two languages, denoted by $L_1/L_2$ is given by

$$L_1/L_2 = \{x \mid (\exists y \in L_2)[xy \in L_1]\}.$$ 

Proposition. If $L_1$ is a regular language, then $L_1/L_2$ is regular.

Proof. Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA accepting $L_1$. By definition, for any $x \in L_1/L_2$ $\exists y \in L_2$ s.t. $\delta(q_0, xy) = \delta(\delta(q_0, x), y) \in F$. If we define

$$F' = \{q \in Q \mid (\exists y \in L_2)[\delta(q, y) \in F]\},$$

$M' = (Q, \Sigma, \delta, q_0, F')$ is a DFA that accepts $L_1/L_2$.

Note: The proof is not quite constructive, at the moment, since we would need to show that a finite number of strings in $L_2$ suffices to construct $F'$. 
Finite Automata

Closure Properties of Regular Languages - Examples

Ex. 2.37. Let \( L \) be regular over \( \Sigma \), \( k \) a pos. integer and \( \phi: \Sigma^k \rightarrow \Sigma \). Then

\[
L_1 = \{ \phi(a_1, a_2, \ldots, a_k) \ldots \phi(a_{(n-1)k}, a_{(n-1)k+1}, \ldots, a_{nk}) \mid a_1a_2\ldots a_{nk} \in L \}
\]

is regular. Note that a string of length \( nk \) in \( L \) becomes a string of length \( n \) in \( L_1 \).

Proof. Let \( L = L(M) \) for \( M = (Q, \Sigma, \delta, s, F) \). Let \( M' = (Q, \Sigma, \delta', s, F) \), with

\[
\forall (q \in Q, a \in \Sigma) \delta'(q, a) = \{ \delta(q, a_1\ldots a_k) \mid \phi(a_1, \ldots, a_k) = a \}.
\]

Note: there is no requirement for \( \phi \) to be 1-1.

We can see that \( L_1 = L(M') \).

Note that \( \delta(q, a_1\ldots a_k) \) is the state of \( M \) reached from \( q \) after consuming the string \( a_1\ldots a_k \).
Closure Properties of Regular Languages - Examples

**Def.** for any language $L$,
\[
\operatorname{MIN}(L) = \{ x \in L \mid \text{no proper prefix of } x \text{ belongs to } L \}.
\]

**Ex. 2.38.** If $L$ is regular, then $\operatorname{MIN}(L)$ is regular.

Proof. Let $L = L(M)$, where $M = (Q, \Sigma, \delta, s, F)$ is a DFA. Let $M'$ be the NFA obtained from $M$ by deleting all the out-edges from the final states.
It is clear that $M'$ accepts $\operatorname{MIN}(L)$…
Closure Properties of Regular Languages - Examples

**Def.:** for \( a, b \in \{0, 1\} \), let \( a \lor b \) denote the disjunction of \( a \) and \( b \); for any two binary strings \( x \) and \( y \), with \( |x| = |y| \), let \( x \lor y \) denote the bitwise disjunction of \( x \) and \( y \).

**Ex. 2.39.** Let \( A \) and \( B \) be regular over \( \{0, 1\} \). Then
\[
A \lor B = \{ x \lor y \mid x \in A, y \in B, |x| = |y| \}
\]
is regular.

**Proof.** Let \( M_A = (Q_A, \{0, 1\}, \delta_A, s_A, F_A) \) and \( M_B = (Q_B, \{0, 1\}, \delta_B, s_B, F_B) \) be the respective DFAs. We build a product NFA \( M' \) as follows:
\[
M' = (Q_A \times Q_B, \{0, 1\}, \delta', [s_A, s_B], F_A \times F_B)
\]
where
\[
\delta'([p, q], 0) = \{[\delta_A(p, 0), \delta_B(q, 0)]\}
\]
\[
\delta'([p, q], 1) = \{[\delta_A(p, 1), \delta_B(q, 1)], [\delta_A(p, 1), \delta_B(q, 0)], [\delta_A(p, 1), \delta_B(q, 1)]\}.
\]
Minimum Deterministic Finite Automata

We have seen that regular languages give rise to at least three different, and equivalent, notational devices that represent them. For one of them, at least, the following question is meaningful: does a regular language have a corresponding (unique) minimal DFA, in the sense that no other DFA accepting the language has fewer states? if the answer is yes, can we construct it?

The constructions we have, from REs to NFAs to DFAs tend to "blow up" the number of states, up to exponential cardinality in the number of states of the NFA - which already "bloats" the number of atomic terms in the regular expression (digraph construction).

We could use a different characterization of regular languages.
Finite Automata

Minimum Deterministic Finite Automata - the Index

Definition. For any language $L \subseteq \Sigma^*$, we define a relation $R_L$ on $\Sigma^*$:

$$x R_L y \iff (\forall w \in \Sigma^*) [xw \in L \iff yw \in L].$$

Proposition. $R_L$ is an equivalence relation on $\Sigma^*$:

1) it is reflexive: $(\forall x \in \Sigma^*) [x R_L x]$
2) it is symmetric: $(\forall x, y \in \Sigma^*) [x R_L y \implies y R_L x]$
3) it is transitive: $(\forall x, y, z \in \Sigma^*) [x R_L y, y R_L z \implies x R_L z]$

Proof. An easy exercise.

Corollary. $R_L$ partitions $\Sigma^*$ into disjoint equivalence classes, where the class containing $x$ is denoted by $[x]_{R_L}$.

Proof. Another easy exercise.

Definition. The number of equivalence classes of $R$ in $\Sigma^*$ is called the index of $R$ and denoted by $\text{Index}(R)$. 
Minimum Deterministic Finite Automata - the Index - Example

Ex. 2.46: Let $L$ be the set of binary strings starting and ending with the same symbol. Find all equivalence classes of $R_L$ (in $\Sigma^*$).

Soln.: a) we first show that $\forall x, y \in \Sigma^*, x R_L y \iff x$ and $y$ start with the same symbol and end with the same symbol.

$\Rightarrow$ By contradiction. Assume $x$ and $y$ start with different symbols, with, say, $x$ starting with a 0 and $y$ starting with a 1. Then $x0 \in L$ while $y0 \notin L$, so that $x R_L y$ does not hold. This implies that $x$ and $y$ must start with the same symbol. Note further that $x R_L y \Rightarrow x \in L \iff y \in L$ ($\epsilon \in \Sigma^*$), and therefore, since they start with the same symbol, they must end with the same symbol.
Finite Automata

Minimum Deterministic Finite Automata - the Index - Example

⇐ Assume $x$ and $y$ start with the same symbol and end with the same symbol. Since they start with the same symbol: $xw \in L \iff yw \in L$ for all $w \in (0+1)^+$. Since $x$ and $y$ end with the same symbol, $x \in L \iff y \in L$. Thus $xw \in L \iff yw \in L$ for any $w \in (0+1)^* = \Sigma^*$.

We can now characterize the (5) equivalence classes:

$[\varepsilon]_{RL} = \varepsilon,$

$[0]_{RL} = 0 + 0(0 + 1)^*0,$

$[01]_{RL} = 0(0 + 1)^*1,$

$[1]_{RL} = 1 + 1(0 + 1)^*1,$

$[10]_{RL} = 1(0 + 1)^*0.$
How do we find the classes of $\Sigma^*$ induced by the relation $R_L$? If we cannot guess them - and we usually can't - the only strategy left is to be systematic.

1. Start from the string $\epsilon \in \Sigma^*$ and find the strings in $\Sigma^*$ that belong to its class.
2. Move to the strings of length 1
3. Move to the strings of length 2 … etc.
4. Either the classes will begin repeating after a finite number of constructions (in which case the index will be finite), or not, in which case the index will be infinite.
Minimum Deterministic Finite Automata - the Index - Example

Details: recall \( x \mathrel{R_L} y \iff (\forall w \in \Sigma^*) \ [xw \in L \iff yw \in L] \), where
\[
L = \varepsilon + 0 + 1 + 0(0+1)^*0 + 1(0+1)^*1
\]

Show: \([\varepsilon]_{R_L} = \{\varepsilon\}\), where
\[
[\varepsilon]_{R_L} \equiv \{ y \mid y \in \Sigma^* \text{ and } (\forall w \in \Sigma^*) \ [\varepsilon w = w \in L \iff yw \in L]\}
\]

Since \(\varepsilon w \in L \iff w \in L\), we are looking for those strings \(y\) s.t. \(yw \in L \iff w \in L\). \(y = \varepsilon\) is, obviously, OK. Let's now start with the shortest non-trivial candidates: \(y = 0, 1\). For each \(w \in L\), \(0w \in L\) if \(w = \varepsilon\) or \(w\) ends in \(0\), otherwise \(0w \not\in L\). So \(0 \not\in [\varepsilon]_{R_L}\). The same argument gives that \(1 \not\in [\varepsilon]_{R_L}\). If \(y\) is any other nontrivial string, it must be one of \(0(0+1)^*0\), \(0(0+1)^*1\), \(1(0+1)^*0\) and \(1(0+1)^*1\). Then \(\varepsilon 11 \in L\) and \(0(0+1)^*011\), \(0(0+1)^*111 \not\in L\). Similarly for \(\varepsilon 00\) and \(1(0+1)^*000\), \(1(0+1)^*100\).

So \([\varepsilon]_{R_L}\) contains just one string: \(\varepsilon\).
We now look at the classes of longer strings. The strings of length 1 will give rise to the classes \([0] R_L\) and \([1] R_L\).

**Claim:** \([0] R_L = 0 + 0(0 + 1)^*0\), \([1] R_L = 1 + 1(0 + 1)^*1\).

Let's construct \([0] R_L\), the construction of \([1] R_L\) being similar.

\[ [0] R_L \equiv \{ y \mid y \in \Sigma^* \text{ and } (\forall w \in \Sigma^*) [0w = w \in L \iff yw \in L] \]

Obviously, \(0 \in [0] R_L\). The rest of the candidates must be of the form \(y = 1, 0(0 + 1)^*0, 0(0 + 1)^*1, 1(0 + 1)^*0\) and \(1(0 + 1)^*1\). For \(y = 1\), the choice \(w = 1\) will invalidate the condition; for \(y = 0(0 + 1)^*0\), the condition will hold for \(w = \varepsilon\) (both 0 and \(0(0+1)^*0\) begin and end in 0) and for all other choices of \(w\) it will depend only on the last character of \(w\) being 0 or not. For \(y\) having the form of one the three remaining string types, we find failures for \(w = \varepsilon\) in the cases \(0(0 + 1)^*1\) and \(1(0 + 1)^*0\) and for \(w = 0\) in the last one.
Minimum Deterministic Finite Automata - the Index - Example

We still have to place two sets of strings: the ones denoted by $0(0 + 1)^*1$ and $1(0 + 1)^*0$. It should be clear that, after this placement, \textbf{all} strings in $\Sigma^*$ will have been taken care of.

An argument similar to those already encountered allows us to conclude that:

\[
[01]_{RL} = 0(0 + 1)^*1, \quad [10]_{RL} = 1(0 + 1)^*0
\]
Problem 2.7.1a: find all equivalence classes of $R_L$ for the language $L = (0 + 1)^*01(0 + 1)^*$.

**Soln:** first observe that $\Sigma = \{0, 1\}$. $R_L$ is defined over $\Sigma^* (= \text{all strings over } \{0, 1\})$ - we are partitioning the set of all strings over $\{0, 1\}$ and, specifically,

$$x R_L y \iff \forall (w \in \Sigma^*)\{xw \in L \iff yw \in L\}$$

for any two strings $x, y \in \Sigma^*$.

Start with the class of the empty string: $[\varepsilon]_{R_L}$.

**Claim:** $[\varepsilon]_{R_L} = [1]_{R_L} = [11]_{R_L} = \ldots = [1^n]_{R_L} = \ldots$

Note that both $\varepsilon$ and $1^n$ must be concatenated with a string containing 01 (thus in $L$) in order to give rise to a string in $L$. 
Problem 2.7.1a: find all equivalence classes of $R_L$ for the language

$L = (0 + 1)^*01(0 + 1)^*$.

Claim: $[0]_{RL} = \ldots = [0^n]_{RL} = \ldots = [1^m0^n]_{RL}$ for $m \geq 1$, $n \geq 1$, i.e., $w$ must contain a 1 for the concatenation to be in $L$.

Claim: $[1]_{RL} = [\varepsilon]_{R_I}$ from the previous slide.

Claim: $[01]_{RL} = L$.

All other strings in $\Sigma^*$ belong to one of the classes just constructed.

Automaton for $L$:
Problem 2.7.2a: show that for \( L = \{0^m1^n \mid 0 \leq m \leq n\} \), \( \text{Index}(L) = \infty \).

**Soln:** first observe that \( \Sigma = \{0, 1\} \). \( R_L \) is defined over \( \Sigma^* \) (= all strings over \( \{0, 1\} \) - we are partitioning the set of all strings over \( \{0, 1\} \)) and, specifically,

\[
x R_L y \iff \forall (w \in \Sigma^*)\{xw \in L \iff yw \in L\}
\]

for any two strings \( x, y \in \Sigma^* \).

**Start** with the class of the empty string: \([\varepsilon]_{R_L} : \) for what \( y \in \Sigma^* \) is it true that \( \varepsilon w \in L \iff yw \in L \)? It is clear that \( \varepsilon w \in L \iff w \in L \), so that \( w = 0^m1^n \) for some \( 0 \leq m \leq n \).

**Question:** what must \( y \) look like so that \( yw \in L \) for all \( w \) of the form \( 0^m1^n \), \( 0 \leq m \leq n \), and only for those \( w \)s? Assume \( y = 0^k \) for some \( k > 0 \). Then \( 0^k0^m1^n \notin L \forall (0 \leq m \leq n < k + m) \). Similar arguments hold for any non-empty strings: \([\varepsilon]_{R_L} = \{\varepsilon\} \).
Finite Automata

Minimum DFAs - the Index - Third Example

Let's now try $x = 0^k, k > 0$. Let $y = 0^i, j \neq k$. We must satisfy the requirement $0^k w \in L \iff 0^i w \in L \ \forall (w \in \Sigma^*)$. If we choose $l = \min\{j, k\}$, then either $0^k 1^l$ or $0^k 1^l \not\in L$.

Since we have just shown that $0^k$ and $0^i, j \neq k$, belong to different equivalence classes, we must conclude that $R_L$ has infinitely many equivalence classes.

Notice that we do not need to identify all the distinct equivalence classes - there may be others, after all we have identified only $[\varepsilon]_{RL}, [0]_{RL}, [0^2]_{RL}, \ldots, [0^n]_{RL}, \ldots$ but this is enough to reach the desired conclusion.
Finite Automata

Minimum Deterministic Finite Automata

Lemma. Let $L$ be accepted by the DFA $M = (Q, \Sigma, \delta, q_0, F)$, and let $\delta$ be the extended transition function of $M$. Then, for any strings $x, y \in \Sigma^*$, $\delta(s, x) = \delta(s, y) \Rightarrow x \sim_L y$.

Proof. If $\delta(s, x) = \delta(s, y)$, then, for any $w \in \Sigma^*$,

$$\delta(s, xw) = \delta(\delta(s, x), w) = \delta(\delta(s, y), w) = \delta(s, yw).$$

Thus, for any $w \in \Sigma^*$, $xw \in L \Leftrightarrow yw \in L$.

Note. The lemma shows that, if $L$ is accepted by a DFA $M$ of $n$ states, all strings with the same ending state are in the same equivalence class of $\sim_L$. In particular $\text{Index}(\sim_L) \leq n$, and is thus finite. If we can find a DFA accepting $L$ and with exactly $\text{Index}(\sim_L)$ states, we have found the desired minimum DFA.
Finite Automata

Minimum Deterministic Finite Automata

Theorem 2.48 (Myhill-Nerode). For any regular language $L$, its minimum DFA has exactly $\text{Index}(R_L)$ states.

Proof. Let $L$ have alphabet $\Sigma$. Define $M = (Q, \Sigma, \delta, s, F)$ by:

1. $Q = \{[x]_{RL} \mid x \in \Sigma^*\}$, (and $|Q| < \infty$ if $L$ is regular);
2. $\delta([x]_{RL}, a) = [xa]_{RL}$, for any $a \in \Sigma$;
3. $s = [\varepsilon]_{RL}$;
4. $F = \{[x]_{RL} \mid x \in L\}.$

The function $\delta$ is well-defined since:

$$[x]_{RL} = [y]_{RL} \implies [xa]_{RL} = [ya]_{RL} \implies \delta([x]_{RL}, a) = [xa]_{RL} = [ya]_{RL} = \delta([y]_{RL}, a).$$

By induction on $|y|$ we extend this to $\delta([\varepsilon]_{RL}, x) = [x]_{RL} \forall x \in \Sigma^*$, proving that $L(M) = L: x \in L \iff [x]_{RL} \in F \iff \delta([\varepsilon]_{RL}, x) \in F \iff M$ accepts $x$. Since $M$ has $\text{Index}(R_L)$ states, it is, by the previous Lemma, a min. DFA for $L$. 
Finite Automata

Minimum Deterministic Finite Automata

Corollary. A language $L$ is regular $\iff$ $\text{Index}(R_L) < \infty$.

Proof. No use of regularity (other than for the finiteness of $Q$) was made in the Myhill-Nerode theorem. We can thus construct the minimum DFA for any language as along as $\text{Index}(R_L) < \infty$. Such languages $L$ must be regular.

Note: the finiteness of DFAs allows us to obtain a "simplification" - we don't need to look at all of $\Sigma^*$ to characterize $R_L$, but only to strings up to a certain length.
Finite Automata

Minimum Deterministic Finite Automata

Proposition. $L$ is regular $\iff \exists k \in \mathbb{N}^+ \text{ s.t. } [x R_L y \iff [\forall z \in \Sigma^* \text{ with } |z| \leq k, xz \in L \iff yz \in L]]$.

This simply means that regular languages are really determined by finite sets of strings: you will have no surprises from "very long strings".

This also means, in the language quotient case, that we don't need to look at all the strings in $L^2$. 
Finite Automata

Minimum Deterministic Finite Automata

Proposition. \( L \) is regular \( \iff \exists k \in \mathbb{N}^+ \) s.t. \([x R_L y \iff \forall z \in \Sigma^* \text{ with } |z| \leq k, \ xz \in L \iff yz \in L]\).

Proof. \( \Rightarrow \) Let \( M = (Q, \Sigma, \delta, s, F) \) be a DFA accepting \( L \); let \( k = |Q|^2 - 1 \).

\( \Rightarrow \) This is obvious, since \( x R_L y \Rightarrow [\forall z \in \Sigma^* \ xz \in L \iff yz \in L] \), and therefore for all those \( z \) with \( |z| \leq k \).

\( \Leftarrow \) Consider the product DFA \( M^* = M \times M \). Let \( w \in \Sigma^* \) with \( |w| > k \). Let \( x, y \in \Sigma^* \). The computation path of \( M^* \) on \( w \) starting from \( q_1^* = [\delta(s, x), \delta(s, y)] \), to \( q_2^* = [\delta(s, xw), \delta(s, yw)] \), contains at least \( |Q|^2 + 1 \) states, while \( M^* \) has \( |Q|^2 \) states, and therefore it contains some cycles in the transition diagram of \( M^* \). Eliminate the cycles and keep only a simple path from \( q_1^* \) to \( q_2^* \), which corresponds to the computation path for a string \( z \) with \( |z| \leq k \), so that \( q_2^* = [\delta(s, xz), \delta(s, yz)] = [\delta(s, xw), \delta(s, yw)] \). By assumption, \( xz \in L \iff yz \in L \) : the states \( \delta(s, xz) \) and \( \delta(s, yz) \) are both in \( F \) or not in \( F \). So \( xw \in L \iff yw \in L \) and the result follows.
Finite Automata

Minimum Deterministic Finite Automata

\[ \iff \exists k \in \mathbb{N}^+ \text{ s.t. } [x R_L y \iff \forall z \in \Sigma^* \text{ with } |z| \leq k, xz \in L \iff yz \in L]. \]

Each \( z \) can divide \( \Sigma^* \) in at most two parts: \( \{x \mid xz \in L\} \) and \( \{x \mid xz \notin L\} \). The number of equivalence classes of \( R_L \) is thus, at most, \( 2^{1+|\Sigma|+...+|\Sigma|^k} \), and so finite. Which implies regularity for \( L \).

**Note:** we have developed some results and techniques which allow us to do two things

a) we have a way of constructing a minimum DFA for a regular language;

b) we have a way of determining (maybe) whether a language is regular or not (being unable to construct a DFA is quite different from proving that no DFA exists) by computing \( \text{Index}(R_L) \).
Minimum Deterministic Finite Automata - Example

Ex. 2.51: find the minimum DFA for the language \( L = (0 + 1)^*01 \).

Soln.: we compute the equivalence classes of \( R_L \) (some "bit analysis"), beginning from the shortest string in \( \Sigma^* \):

1. \([\varepsilon]_{R_L} = \{x \in \{0, 1\}^* | (\forall w \in \{0, 1\}^*)[xw \in (0 + 1)^*01 \iff w \in (0 + 1)^*01]\}.\) We need to characterize \( x \in [\varepsilon]_{R_L} \). What are the possible relevant endings of \( x \)? : 0, 1, 00, 01, 10, 11. There is no point in looking at longer strings, since \( L \) fixes the last two characters. \( x \) may not end with 0 (or 01): if it did \( x1 \) (or \( x\varepsilon \)) would be in \((0+1)^*01\), which would imply that \( w = 1 \) (or \( w = \varepsilon \)) is in \((0 + 1)^*01\). Conversely, if \( x \) does not end with 0 or 01, then \( x \in [\varepsilon]_{R_L} \). Thus \([\varepsilon]_{R_L} = \varepsilon + 1 + (0 + 1)^*11\).

2. \([0]_{R_L} = \{x \in \{0, 1\}^* | (\forall w \in \{0, 1\}^*)[0w \in (0 + 1)^*01 \iff xw \in (0 + 1)^*01]\}\). What must \( x \) end with? Since, for \( w = 1, 0w \in (0 + 1)^*01\}, \( x \) must end with 0. Conversely, if \( x \) ends with 0, \( x \in [0]_{R_L} \). Thus \([0]_{R_L} = (0 + 1)^*0\).
Finite Automata

Minimum Deterministic Finite Automata - Example

3. \([1]_{RL}\). Using the definition of \([\varepsilon]_{RL}\) we can see that \(1 \in [\varepsilon]_{RL}\) and thus the two classes must coincide (by \(R_L\) being an equivalence relation).

4. \([00]_{RL}\). \(00 \in [0]_{RL} \Rightarrow [00]_{RL} = [0]_{RL}\).

5. \([01]_{RL} = \{x \in \{0, 1\}^+ | (\forall w \in \{0, 1\}^* [01w \in (0 + 1)^*01 \iff xw \in (0 + 1)^*01]\}\). Either \(x\) ends in \(01\), or it cannot satisfy the condition. We must conclude that \([01]_{RL} = (0+1)^*01\).

6. \([10]_{RL}\). since \(10 \in [0]_{RL}\), we have \([10]_{RL} = [0]_{RL}\).

7. \([11]_{RL}\). since \(11 \in [\varepsilon]_{RL}\), we have \([11]_{RL} = [\varepsilon]_{RL}\).

The classes are \([\varepsilon]_{RL}\), \([0]_{RL}\), and \([01]_{RL}\). The minimum DFA has only 3 states:
Minimum Deterministic Finite Automata - Example

Ex. 2.53. Find the minimum DFA equivalent to the DFA:
Minimum Deterministic Finite Automata - Example

Soln. 1: \( M = (Q, \Sigma, \delta, q_0, F) \), \( Q = \{q_0, q_1, q_2, q_3, q_4, q_5\} \), \( L = L(M) \).

Recall: for states \( p \) and \( q \),

\[ p R^* L q \iff S_p = \{x \in \Sigma^* \mid \delta(s, x) = p\} \text{ is in same equiv. class as} \]
\[ S_q = \{x \in \Sigma^* \mid \delta(s, x) = q\} \].
\[ \iff (\forall w)[\delta(p, w) \in F \iff \delta(q, w) \in F]. \]

To find \( R^*_L \) between any two states, construct a graph \( G \):

Each vertex is an unordered pair \((q_i, q_j)\). Let \( U \) be the set of vertices \((q_i, q_j)\) with one vertex \( q_i \in F \) and the other vertex \( q_j \notin F \). For each vertex \((q_i, q_j) \notin U\) (either both vertices are in \( F \) or not in \( F \)), with \( i \neq j \), draw edges \((q_i, q_j) \rightarrow^a (\delta(q_i, a), \delta(q_j, a)) \forall a \in \Sigma\).

Claim: \( q_i R^*_L q_j \iff \) there is no path in \( G \) from \((q_i, q_j)\) to a vertex in \( U \).
Finite Automata

Minimum Deterministic Finite Automata - Example

How do we use this construction? Construct G.

U consists of the pairs (0, 2), (1, 3), (1, 2), (0, 3), (2, 4), (3, 4), (4, 5).

The complement of U consists of (0, 0), (0, 1), (1, 1), (0, 4) (1, 4), (2, 2),
(2, 3), (3, 3), (2, 5), (3, 5), (4, 4), (5, 5).

Note: there is no point in starting from a \((q_i, q_i)\) node - why?

From (0, 1):

![Diagram showing transitions from (0, 1) to (2, 3) and (4, 4) for the automaton G.](attachment:diagram.png)
Finite Automata

Minimum Deterministic Finite Automata - Example

The next node in the list is (0, 4):

Note that from the nodes (0, 4), (1, 4), (2, 5) and (3, 5) we can reach a node in U. When we put together the two partial graphs (constructed this way for convenience, we get:
Minimum Deterministic Finite Automata - Example

The minimal automaton is exactly the one corresponding to the graph with only black nodes:

![Finite Automata Diagram](image)
Minimum Deterministic Finite Automata - Example

We finish with just the minimal DFA, where we have highlighted it final nodes:
Finite Automata

Minimum Deterministic Finite Automata - Example

Soln. 2: The critical step of soln. 1 is to determine, for a pair \((q_i, q_j)\) whether there is a path in \(G\) from it to a vertex of \(U\). You need only study pairs where \(i \neq j\), since you know the \((q_i, q_i)\) belong to the same class. Create a table of pairs, marking all pairs in \(U\) with a 0:

Of the unmarked pairs, mark with a 1 those for which there is an \(a \in \Sigma\) taking them to a pair marked with a 0.
Minimum Deterministic Finite Automata - Example

Of the unmarked pairs, mark with a 2 those for which there is an $a \in \Sigma$ taking them to a pair marked with a 1. Keep on until no more markings. The pairs left unmarked must satisfy $q_i R^*_L q_j$. Add the $(q_i, q_i)$ needed to complete the graph.
Minimum Deterministic Finite Automata - Example

Soln. 3: Split states into F and Q\F. Chase states from F: if any exit F, break F into an appropriate number of sub-blocks; same for Q\F. Keep on iterating until no new blocks appear.
Finite Automata

Minimum Deterministic Finite Automata - Example