Fundamentele Informatica II

Answer to selected exercises 2

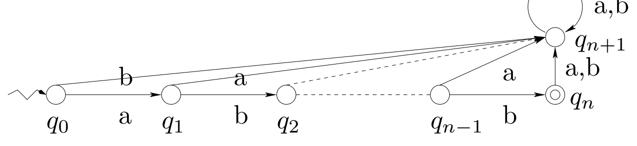
John C Martin: Introduction to Languages and the Theory of Computation

M.M. Bonsangue (and J. Kleijn)

Fall 2013

2.2
(a) All strings containing a substring starting with at least two a's followed by ba.
(b) All strings ending with at least two a's followed by ba.
(c) All strings starting with $aaba$.
(d) Either the empty string or all string starting with a and ending with b.
(d) All strings with no two consecutive equal alphabet symbols (thus no aa or bb).

2.3 b Let $x \in \{a, b\}^*$ be such that $|x| = n$ for some $n \geq 0$. Thus $x = a_1 a_2 \ldots a_n$ with each $a_i \in \{a, b\}$. Then there is a finite automaton accepting $\{x\}$ with $n + 2$ states, namely initial state q_0, intermediate states $q_1, \ldots, q_{n-1}$, accepting state q_n, and sink state q_{n+1}. Observe that for $x = \Lambda$ we have $q_0 = q_n$ as both the initial and the accepting state. Each non-sink state q_i represents the prefix of length i of x and has a transition to the state representing the next longer prefix and — for the "wrong" input symbol a transition to the sink state; from the state q_n representing x and from the sink itself there are only transitions to the sink. Below is an example for a word beginning with ab and with b as its last symbol.



Observe that $n + 2$ states are necessary because different prefixes need different states: if $x = yz$ and $x = uv$ with y and u two different prefixes of x, then $uz \neq yz = x$ and hence uz should not be accepted, which implies that

u should be distinguished from y. That is, y and u should lead to different states. Since x has $n+1$ different prefixes this implies that we need at least $n+1$ states in any finite automaton accepting $\{x\}$. Moreover, all prefixes have to be distinguished from all non-prefixes of x since no word starting with a non-prefix of x will ever lead to acceptance.

2.5 Let $M = (Q, \Sigma, q_0, A, \delta)$ be an FA. We prove by induction on $|y|$ that for all $x, y \in \Sigma^*$ and for all $q \in Q$,
$\delta^*(q, xy) = \delta^*(\delta^*(q, x), y)$.
Let $x \in \Sigma^*$ be an arbitrary word.
basis: $|y| = 0$, that is, $y = \Lambda$. Since, by definition $\delta^*(p, \Lambda) = p$ for all states p, it follows that $\delta^*(q, xy) = \delta^*(q, x) = \delta^*(\delta^*(q, x), \Lambda) = \delta^*(\delta^*(q, x), y)$ as required.
induction hypothesis: there is a $k \geq 0$ such that for all $z \in \Sigma^*$ with $|z| \leq k$, and for all $q \in Q$, $\delta^*(q, xz) = \delta^*(\delta^*(q, x), z)$.
induction step: let $|y| = k+1$, that is, $y = za$ for some $a \in \Sigma$ and $z \in \Sigma^*$ such that $|z| = k$.
We now have $\delta^*(q, xy) = \delta^*(q, xza) = \delta(\delta^*(q, xz), a)$, by the definition of δ^*.
With the induction hypothesis we obtain $\delta(\delta^*(q, xz), a) = \delta(\delta^*(\delta^*(q, x), z), a)$ which — again by the definition of δ^* — equals $\delta^*(\delta^*(q, x), za) = \delta^*(\delta^*(q, x), y)$.
Our claim now follows for the chosen x, but since that was an arbitrary word, the statement has been proved for all x.
Using induction on $|x|$ rather than on $|y|$ would have been awkward (or even impossible) given the structure of the (inductive) definition of δ^*.

2.6 Let $M = (Q, \Sigma, q_0, A, \delta)$ be an FA. Let $q \in Q$ be such that $\delta(q, a) = q$ for all $a \in \Sigma$. This means that whatever the input symbol, if M is in state q it will remain there. We use induction to prove that once in q, M will never leave that state anymore whatever the input string it is being fed.
By definition, we have that $\delta((q, \Lambda) = q$.
Now assume that there is an integer $k \geq 0$ such that $\delta^*(q, z) = q$, for all $z \in \Sigma^*$ with $|z| \leq k$. Let $x \in \Sigma^*$ be of length $k+1$. So, $x = za$ for some $a \in \Sigma$ and $z \in \Sigma^*$ with $|z| = k$. Consequently, we can use the induction hypothesis and deduce that $\delta^*(q, x) = \delta^*(q, za) = \delta(\delta^*(q, z), a) = \delta(q, a) = q$ by the property given for q.

2.10 a., b. and c. The desired automata each have (A, X) as initial state. Next we apply the product construction to M_1 and M_2:

	a	b
(A,X)	(B,X)	(A,Y)
(B,X)	(B,X)	(C,Y)
(A,Y)	(B,X)	(A,Z)
(C,Y)	(B,X)	(A,Z)
(A,Z)	(B,Z)	(A,Z)
(B,Z)	(B,Z)	(C,Z)
(C,Z)	(B,Z)	(A,Z)

Note that the states (B,Y) and (C,X) which are not reachable from (A,X) are not mentioned in the table for the product transition function (see exercise 3.29).
For $L_1 \cup L_2$, we have as accepting states $\{(C,Y),(C,Z),(A,Z),(B,Z)\}$, that is any pair of original states in which at least one is accepting.
For $L_1 \cap L_2$, we have as accepting states $\{(C,Z)\}$, that is any pair of original states in which both are accepting.
For $L_1 - L_2$, we have as accepting states $\{(C,Y)\}$, that is any pair of original states in which the first is accepting and the second not.
Drawing the automata is now easy.

2.11 Let $M_1 = (Q_1, \Sigma, q_1, A_1, \delta_1)$ be an FA and let $M_2 = (Q_2, \Sigma, q_2, A_2, \delta_2)$. Define for all $p \in Q_1$, $q \in Q_2$, and $a \in \Sigma$: $\delta((p,q),a) = (\delta_1(p,a), \delta_2(q,a))$.
We have to prove that $\delta^*((p,q),x) = (\delta_1^*(p,x), \delta_2^*(q,x))$ for all $p \in Q_1$, $q \in Q_2$, and $x \in \Sigma^*$ (see the proof of Theorem 2.15).
We use induction on $|x|$, the length of x.
If $|x| = 0$, then $x = \Lambda$ and we have $\delta^*((p,q),\Lambda) = (p,q) = (\delta_1^*(p,\Lambda), \delta_2^*(q,\Lambda))$ by the inductive definitions of δ^*, δ_1^*, and δ_2^*.
Let $|x| = n+1$. Then $x = ya$ for some $y \in \Sigma^*$ and $a \in \Sigma$. Hence $|y| = n$ and according to the induction hypothesis: $\delta^*((p,q),y) = (\delta_1^*(p,y), \delta_2^*(q,y))$.
So, $\delta^*((p,q),x) = \delta^*((p,q),ya) = \delta(\delta^*((p,q),y)a)$ by the inductive definition of δ^*. By the induction hypothesis $\delta(\delta^*((p,q),y),a) = \delta((\delta_1^*(p,y), \delta_2^*(q,y)),a)$ and $\delta((\delta_1^*(p,y), \delta_2^*(q,y)),a) = (\delta_1((\delta_1^*(p,y),a), \delta_2((\delta_2^*(q,y),a))$ by the definition of δ. Finally, using the inductive definition of δ_1^* and δ_2^*, we derive $(\delta_1((\delta_1^*(p,y),a), \delta_2((\delta_2^*(q,y),a)) = (\delta_1^*(p,ya), \delta_2^*(q,ya)) = (\delta_1^*(p,x), \delta_2^*(q,x))$ as desired.

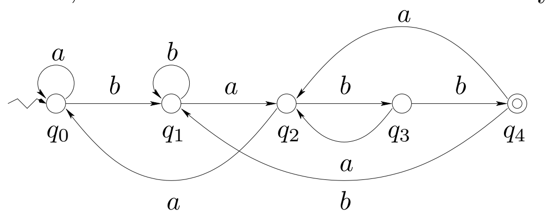
• *Show by an example that for some language L, any FA recognizing L must have more than one accepting state. Characterize those languages for which this is true.*
Let $L = \{\Lambda, 0\} \subseteq \{0\}^*$. Observe that Λ and 0 are distinguishable with re-

spect to L, since they are distinguished by, e.g., the word $z = 0$: $\Lambda 0 \in L$ but 00 is not a word from L. Thus any FA accepting L has two distinct states (see Lemma 3.1.) for Λ and 0. Since both words are in L, these states are both accepting.
In general, for every language which contains at least two distinguishable words, any FA recognizing that language has at least two accepting states. This can — similar to the above — be seen as follows: Let M be an arbitrary FA accepting L and let $u, v \in L$ be distinguishable with respect to L. Then by Lemma 3.1, $\delta^*(q_0, u) \neq \delta^*(q_0, v)$ where q_0 is the initial state of M. Consequently, M has at least two accepting states.
Also the converse holds: for every language L that can be accepted by an FA, it is the case that if no FA accepting L has less than two accepting states, then L contains two words which are distinguishable with respect to L. This can be proved once it has been shown that for every language that can be accepted by an FA, there exists a minimal FA: two strings are indistinguishable w.r.t. the language if and only if they lead to the same state in that FA.

2.13 Consider the FA from Figure 2.17d. It accepts the language L consisting of all strings from $\{a, b\}^*$ that do not contain aa and do not end in ab. The simplest strings corresponding to its four states are Λ, a, ab, and aa. If any two of these strings are distinguishable, then it follows from Theorem 2.21 that any FA recognizing L has at least 4 states.
for Λ and a and for ab and aa, choose $z = a$:
$\Lambda a \in L$, $aba \in L$, but $aa \notin L$ and $aaa \notin L$;
for Λ and ab, for Λ and aa, for a and ab, and for a and aa, choose $z = \Lambda$:
$\Lambda\Lambda \in L$, $a\Lambda \in L$, but $ab\Lambda \notin L$ and $aa\Lambda \notin L$.
Hence, there is no FA accepting L with fewer states than the given FA.

2.14 Let z be a word over the alphabet $\{a, b\}$. Any FA which accepts $L = \{a, b\}^*\{z\}$ has at least $|z|+1$ states. The reason is that z has $|z|+1$ prefixes (from Λ to z itself) which all have to be distinguished in the automaton. If $z = xy$ and $z = uv$ with $|x| > |u|$, then $xy \in L$, but $uy \notin L$. Hence x and u are distinguishable with respect to L.
No more states are needed, since there is no need to distinguish between a word and the longest prefix of z which is a suffix of this word:
Consider a word x and let v be its longest suffix such that $x = uv$ where v is such that $z = vw$ for some w. (Note that *every* word x has such a suffix!) Then for all words $y \in \{0, 1\}^*$ we have
case $|y| \geq |z|$: $xy \in L$ if and only if $vy \in L$ because it is only relevant

whether y ends with z or not;
or case $|y| < |z|$: then $xy \in L$ if and only if $uvy \in L$ if and only if vy ends with z if and only if $vy \in L$.
Consequently, x and v are indistinguishable and we know how to define an FA accepting L.
As an example we give an FA accepting $\{a,b\}^*\{babb\}$ with 5 states:
q_0 corresponding to matching suffix Λ; q_1 for b; q_2 for ba; q_3 for bab; and q_4 for babb, this is the final state since its last symbols form $z = babb$.



2.15 Let $L = \{aa\}^*$, the language of all even length a-strings. Then we have infinitely many pairs of distinguishable strings: for all $i \geq 0$, the string a^{2i} and the string a^{2i+1} are distinguishable with respect to L.
Note that the language L is accepted by a FA (as is easy to see). In fact, there are only two types of indistinguishable strings with respect to L (the even length strings versus the odd length strings).

2.16 Let $n \geq 0$. Let K be an arbitrary non-empty language consisting of words of length n only. Then any FA that accepts K has at least $n+2$ states which can be seen as follows:
First, we observe that if $n = 0$, then it must be the case that $K = \Lambda$ and any FA accepting K has 2 states (the initial state which is also the only accepting state, and a sink state).
Next, we assume that $n \geq 1$ and we consider a (fixed) word $x \in K$. Since $K \neq \emptyset$, such x exists. This x has (at least) two different prefixes $x_1 \neq x_2$, thus: $x = x_1y_1 = x_2y_2$ for some y_1, y_2. Obviously, $x_2y_1 \notin K$ and so x_1 and x_2 can be distinguished w.r.t. K. Moreover (the infinitely many) words which are not a prefix of a word in K can be distinguished from the $n+1$ prefixes of x. Consequently, by Theorem 2.21, any FA accepting K has at least $n+2$ states.
We conclude that for any infinite language $L \subseteq \{a,b\}^*$ any FA which accepts its subset $L_n = \{x \in L \mid |x| = n\}$ has at least $n+2$ states ($s(n) = 2$). The language $L = \{a,b\}^*$ has the property that each of its L_n's can be recognized by an FA with exactly $s(n) = n+2$ states. Any language that does not contain two strings of the same length also has this property.

2.17 Let $L = \{a^n b^n \mid n \geq 0\}$.
a. Give strings $x, y \in \{a, b\}^*$ such that $x \neq y$ and xI_Ly (that is x and y are indistinguishable w.r.t. L).
$x = a$ and $y = ba$: both are words not in L and both are not prefixes of any word in L. This means that for all $z \in \{a, b\}^*$ we have $xz \notin L$ and $yz \notin L$. In general: every pair of words which both are not a prefix of a word in L are indistinguishable.
$x = ab$ and $y = aabb$: both are in L and any non-empty word concatenated to them yields a word not in L. This means that $x\Lambda \in L$ and $y\Lambda \in L$ and $xz \notin L$ and $yz \notin L$ for all non-empty words z. Hence, for all $z \in \{a, b\}^*$ we have $xz \in L$ if and only if $yz \in L$ and so x and y are indistinguishable with respect to L. In general: every pair of non-empty words of L are indistinguishable with respect to L. Note that $\Lambda \in L$ can be distinguished from $x = ab$, because for example $\Lambda a^3b^3 \in L$ but $aba^3b^3 \notin L$.
$x = aab$ and $y = a^3b^2$: both are not in L, adding a single b gives for each a word in L, while all other words will lead to a word not in L. Hence we have $x\Lambda \notin L$ and $y\Lambda \notin L$, $xb \in L$ and $yb \in L$, and $xz \notin L$ and $yz \notin L$ for all $z \in \{a, b\}^*$ with $z \neq \Lambda$ and $z \neq b$. Consequently, for all $z \in \{a, b\}^*$ we have $xz \in L$ if and only if $yz \in L$. In general: every pair $x = a^m b^{m-k}$ and $y = a^n b^{n-k}$ with $n \neq m$ and $n > k$ or $m > k$ are indistinguishable with respect to L.
b. Consider a^m and a^n with $m \neq n$. These words can be distinguished w.r.t. L (that is $a^m I_L a^n$ does not hold), because $a^m b^m \in L$, but $a^n b^m \notin L$. From Theorem 2.26 we conclude that any finite automaton recognizing L needs a separate state $\delta^*(q_0, a^k)$ for all $k \geq 0$, a contradiction with an FA having only finitely many states. Thus we conclude that L cannot be recognized by a FA.

• *Let $M = (Q, \Sigma, q_0, A, \delta)$ be an FA and let $M_1 = (R, \Sigma, q_0, A \cap R, \delta_1)$ be the FA obtained from M by deleting those states $q \in Q$ that are not reachable from q_0 together with all transitions leading to and from them (i.e there is no string $x \in \Sigma^*$ such that $\delta^*(q_0, x) = q$) Show that $L(M_1) = L(M)$.*
Take $R = \{p \in Q \mid \exists x \in \Sigma^*$ such that $\delta^*(q_0, x) = p\}$. It consists of all states in Q reachable from q_0. Let δ_1 is the restriction of δ to R, that is, $\delta_1(p, a) = \delta(p, a)$ for all $p \in R$ and all $a \in \Sigma$.
Since M_1 is a "sub-automaton" of M it follows immediately that $L(M_1) \subseteq L(M)$. To prove the converse inclusion we consider an arbitrary word $w \in L(M)$. Let $p \in A$ be the accepting state such that $\delta^*(q_0, w) = p$. Hence p is reachable: $p \in A \cap R$. Moreover, every state $s \in Q$ such that $\delta{*}(q_0, v) = s$ for some prefix of w, is reachable from q_0 and hence in R and all its transitions

are in δ_1. Consequently, $\delta^*(q_0, w) = \delta_1^*(q_0, w) = p \in A \cap R$ which implies that $w \in L(M_1)$.

2.27
a. Let M_1 en M_2 be two FA's over an input alphabet Σ.
Are there words in Σ that are not accepted by neither of the two FA's is the same of saying: Is $\Sigma^* - (L(M_1) \cup L(M_2)) = \emptyset$?
Algoritme: construeer een FA M_3 met $L(M_3) = L(M_1) \cup L(M_2)$ (zie bijvoorbeeld de productconstructie in het bewijs van Theorem 3.4) en vervolgens een FA M_4 voor het complement $\Sigma^* - L(M_3)$ (zie weer het bewijs van Theorem 3.4). Gebruik het algoritme op pagina 187 om te bepalen of $L(M_4)$ leeg is.
g. Gegeven een FA M met invoeralfabet Σ en twee strings $x, y \in \Sigma^*$.
Zijn x en y onderscheidbaar m.b.t. $L(M)$?
Algoritme: Minimaliseer M zoals beschreven in 5.2. De resulterende FA $M_1 = (Q, \Sigma, q_0, A, \delta)$ accepteert L. Volg in M_1 vanaf q_0 het pad gelabeld met x en ook het pad gelabeld met y en kijk of ze naar dezelfde toestand leiden. Zoja: x en y zijn niet te onderscheiden m.b.t. L; zo nee, x en y zijn wel te onderscheiden m.b.t. L.
(M_1 heeft als minimale automaat de eigenschap dat $\delta^*(q_0, x) = \delta^*(q_0, y)$ dan en slechts dan als $x \; I_L \; y$).

2.29
a. False: $pal \subseteq \{a, b\}^*$ is a nonregular subset of the regular language $\{a, b\}^*$.
b. False: Nonregular languages have finite subsets, and every finite language is regular.
c. False: The union of any language over an alphabet Σ and its complement is Σ^* which is regular.
d. False: The intersection of any language and its complement is the empty language which is regular.
e. True: The complement of a regular language is always regular (Theorem 3.4). Consequently a language is regular if and only if its complement is regular.
f.: false;
g.: true;
h., **i.**: both false.
j. False: Let for all $i \geq 1$, $L_i = \{a, b\}^* - K_i$ with $K_i = \{a^{2^{i \times n}} b^{2^{i \times n}} \mid n \geq 0\}$.
Thus $K_1 = \{ab, a^2b^2, a^4b^4, a^8b^8, a^{16}b^{16}, \ldots\}$, $K_2 = \{ab, a^4b^4, a^{16}b^{16}, \ldots\}$, $K_3 = \{ab, a^8b^8, a^{64}b^{64}, \ldots\}$ etc.
Clearly, $K_{i+1} \subseteq K_i$ for all $i \geq 1$ which implies that $L_i \subseteq L_{i+1}$ for all $i \geq 1$.

Furthermore it is not too difficult to see that each K_i is not regular, and hence each L_i is not regular.
Note that the shortest word in each K_i is ab and the next shortest word is $a^{2^i}b^{2^i}$. Consequently, $\bigcap_{i=1}^{\infty} K_i = \{ab\}$, a regular language. This implies that $\bigcup_{i=1}^{\infty} L_i = \{a,b\}^* - \bigcap_{i=1}^{\infty} K_i = \{a,b\}^* - \{ab\}$ is also a regular language.

2.32 If $L \subseteq \{a,b\}^*$ consists of one single equivalence class w.r.t. I_L, then are all words in $\{a,b\}^*$ equivalent. Therefore, for every $x, y \in \{a,b\}^*$ it holds that $x = x\Lambda \in L$ if and only if $y = y\Lambda \in L$.
Thus $L = \emptyset$ or $L = \{a,b\}^*$.

2.33 $L = \{x\}$ with x a word over $\{a,b\}$. Then I_L has $|x| + 2$ equivalence classes: one for each prefix of x and one containing all the words that are not a prefix of x.

2.34 Let L be a language over an alphabet Σ and let $x \in \Sigma^*$ be such that it is *not* a prefix of a word in L. For all words xy with $y \in \Sigma^*$, there is no z such that $xyz \in L$. Consequently, the set $S = \{w \in \Sigma^* \mid \text{there exists no } y \text{ such that } wy \in L\}$ is infinite.
We need to prove that S is an equivalence class of I_L: let u, v be two words from S. Then for all words $z \in \Sigma^*$ we have that $uz \notin L$ and $vz \notin L$ since neither u nor v are a prefix of a word in L. Thus, $u\ I_L\ v$. Hence all words in S are indistinguishable with respect to L and so S is contained in a single equivalence class of I_L.
Next let $u \in S$ and $v \notin S$. Then there is a word y such that $vy \in L$, but $uy \notin L$. Hence u and v are distinguishable with respect to L and thus belong to different equivalence classes. Together with the above this implies that S is exactly one infinite equivalence class of I_L.

2.35 Let $L \in \Sigma^*$. If the equivalence class $[\Lambda]$ of I_L contains a word w different from Λ, then for all $z \in \Sigma^*$ we have:
$wz \in L$ if and only if $\Lambda z = z \in L$.
It follows that for all $i \geq 0$ and for all $y \in \Sigma^*$:
$w^{i+1}y = w(w^i y) \in L$ if and only if $w^i y = \Lambda(w^i y) \in L$.
Thus $w^{i+1}\ I_L w^i$ for all $i \geq 0$, that is, $[\Lambda]$ contains $\{w\}^*$ and hence is infinite (because $w \neq \Lambda$).

2.36 $L \subseteq \{a,b\}^*$ is a language for which we are asked to give a finite automaton. We apply the construction used to prove Theorem 5.1.
I_L has 4 equivalence classes $[\Lambda]$, $[a]$, $[ab]$, and $[b]$ which we will use as states. $[\Lambda]$ will be the initial state. Moreover, since $ab \in L$ and $\Lambda, a, b \notin L$, we

designate $[ab]$ as the only final state of the automaton.
The transition function δ is defined as follows.
$\delta([\Lambda], a) = [a]$ and $\delta([\Lambda], b) = [b]$;
$\delta([a], b) = [ab]$ and since $a\ I_L\ aa$, we set $\delta([a], a) = [aa] = [a]$;
similarly, $\delta([ab], a) = [aba] = [b]$ and $\delta([ab], b) = [abb] = [a]$.
What remains are the transitions from $[b]$. We know that b is not a prefix of any word in L. Hence (again using exercise 5.4), $[b]$ is the equivalence class consisting of all words which can never be extended to a word in L. In the automaton this equivalence class is a sink: $\delta([b], a) = [ba] = [b]$ and $\delta([b], b) = [bb] = [b]$.
(Draw the automaton.)

2.39 $L = \{a^n b^n \mid n \geq 0\}$. (See also Example 2.37)
The partition of $\{a, b\}^*$ in equivalence classes of I_L is the same as for the case that $L - \{\Lambda\}$, namely: $\{a^i\}$, for each $i \geq 0$, $\{a^i b^j\}$ for all $i > j > 0$, and $\{x \in \{a, b\}^* \mid \nexists y. xy \in L\}$.

2.40 $L = \{x \in \{a, b\}^* \mid n_a(x) = n_n(x)\}$, the language consisting of all words over $\{a, b\}$ with an equal number of a's and b's.
a. For any two words $x, y \in \{a, b\}^*$ it is the case that $x\ I_L\ y$ whenever the difference between the number of a's and b's in x and y is the same. To prove this statement assume that $n_a(x) - n_b(x) = n_a(y) - n_b(y)$ and consider an arbitrary word $z \in \{a, b\}^*$.
Then $xz \in L$ if and only if
$n_a(xz) - n_b(xz) = 0$ if and only if
$n_a(x) + n_a(z) - (n_b(x) + n_b(z)) = n_a(x) - n_b(x) + n_a(z) - n_b(z) = 0$
if and only if
$n_a(y) - n_b(y) + n_a(z) - n_b(z) = 0$ if and only if
$n_a(yz) - n_b(yz) = 0$ if and only if $yz \in L$.
b. Conversely, for any two words $x, y \in \{a, b\}^*$ it is the case that if $x\ I_L\ y$, then the difference between the number of a's and b's in x and y is the same. To prove this statement we consider two words $x, y \in \{a, b\}^*$ such that $n_a(x) - n_b(x) \neq n_a(y) - n_b(y)$. Now let $z \in \{a, b\}^*$ be such that $n_b(z) - n_a(z) = n_a(x) - n_b(x)$. Consequently, $n_a(xz) - n_b(xz) = n_a(x) + n_a(z) - n_b(x) - n_b(z) = (n_a(x) - n_b(x)) - (n_b(z) - n_a(z)) = 0$. Thus $xz \in L$. However, since $n_a(y) - n_b(y) \neq n_a(x) - n_b(x)$, we know that $n_a(y) - n_b(y) \neq n_b(z) - n_a(z)$ and thus $n_a(yz) - n_b(yz) = (n_a(y) - n_b(y)) - (n_b(z) - n_a(z)) \neq 0$ which implies that $yz \notin L$. Hence $x\ I_L\ y$ does not hold. Summarizing: if $x\ I_L\ y$ it must be the case that $n_a(x) - n_b(x) = n_a(y) - n_b(y)$.
c. From **a.** and **b.** it follows that an equivalence class of I_L containing a

word x is determined by the difference $n_a(x) - n_b(x)$: a string y is in $[x]$ if and only if $n_a(y) - n_b(y) = n_a(x) - n_b(x)$.
This means that we have the following equivalence classes:
$\ldots, [b^k], \ldots, [b^3], [b^2], [b], [\Lambda], [a], [a^2], [a^3], \ldots, [a^k], \ldots,$
where for all $k \geq 0$, the equivalence class $[b^k]$ consists of all words w over $\{a, b\}$ with $n_b(w) - n_a(w) = k$, and the equivalence class $[a^k]$ consists of all words w over $\{a, b\}$ with $n_a(w) - n_b(w) = k$.
Note that I_L has an infinite number of equivalence classes which implies that L is not a regular language.

2.55 See figure 2.45.
a. We construct $S = \{(p, q) \subseteq Q \times Q \mid p \not\equiv q\}$ recursively, following Algorithm 2.40. In the first pass we note that 5 is an accepting state and the other states are not. Thus we mark (with 1) in the $Q \times Q$ table, the entries $(5, 1)$, $(5, 2)$, $(5, 3)$, and $(5, 4)$. (For symmetry reasons it is sufficient to fill in only the lower triangle. At the diagonal, we have the identities (p, p) which are never in S).
In the second pass (given as 2), we find:
$(1, 2) \in S$, because $\delta(1, b) = 3$ and $\delta(2, b) = 5$, and $(3, 5) \in S$;
$(2, 3) \in S$, because $\delta(2, b) = 5$ and $\delta(3, b) = 3$, and $(5, 3) \in S$;
$(1, 4) \in S$, because $\delta(1, b) = 3$ and $\delta(4, b) = 5$, and $(3, 5) \in S$;
$(3, 4) \in S$, because $\delta(3, b) = 3$ and $\delta(4, b) = 5$, and $(3, 5) \in S$;
and no more.
In the third pass, we find no new pairs.

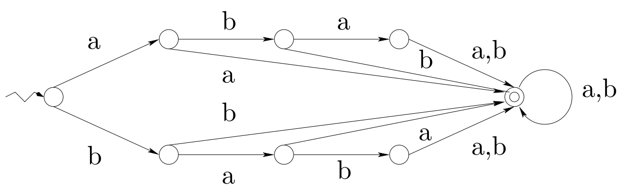
	1	2	3	4	5
1	=				
2	2	=			
3		2	=		
4	2		2	=	
5	1	1	1	1	=

Consequently, we have the following equivalence classes: $\{1, 3\}$, $\{2, 4\}$ and $\{5\}$, which gives a minimal FA with three states $q_0 = \{1, 3\}$, the initial state; $q_1 = \{2, 4\}$; and $q_3 = \{5\}$, the only final state. Its transition function is given in the next table:

	a	b
$q_0 = \{1,3\}$	q_1	q_0
$q_1 = \{2,4\}$	q_0	q_3
$q_3 = \{5\}$	q_1	q_3

Draw this FA.
b. The given FA is already minimal.

2.57 You are asked for a number of languages over the alphabet $\{a, b\}$ to determine whether they can be accepted by an FA or not. If you do well (go after the number of equivalence classes, use the pumping lemma or try to make up an FA for the language) then only given the language in **b.** is regular.
a. Let $L = \{x \in \{a,b\}^* \mid \exists w, y \in \{a,b\}^*.w \neq \Lambda \wedge x = wwy\}$, and take two words $v = ab^n$ and $v' = ab^m$ for $n, m > 0$ and $n \neq m$. For $z = ab^n$ we have that $vz = ab^nab^n \in L$ whereas $v'z = ab^mab^n \notin L$. It follows that all words ab^n for $n > 0$ are pairwise distinguishable, and therefore there can be no FA recognizing L, as it would need infinitely many states by Theorem 2.26.
b. Let $L = \{x \in \{a,b\}^* \mid \exists y, w, z : w \neq \Lambda \wedge x = ywwz\}$.
Claim: $x \in L$ if and only if x contains aa, bb, $abab$ or $baba$ as a subword.
Proof of the claim: if aa, bb, $abab$ of $baba$ a subword is of x, dan is x by definition in L.
Conversely, assume that $x \in L$. Then $|x| \geq 2$. Let aa and bb be not subwords of x. Then in x it holds that every a that is not the last symbol of x, is followed by a b, en that every b that is not the last symbol of x, is followed by an a. Let now $y, w, z \in \{a,b\}^*$ with $w \neq \Lambda$ such that $x = ywwz$. Then we know that $w \neq 0$ and $w \neq 1$; thus either $w = abu$ or $w = bau$. If $u = \Lambda$, then we are done. Otherwise $|u| \geq 2$, and $w = ababv$ or $w = babav$, respectively, en we are ready also in this case.
The only case that remains to check is when $|u| = 1$. If $w = abu$, than must be $u = a$ and $x = ywwz = yabaabaz$, contradicting our assumption that x does not contain aa as subword. Analogously, if $w = bau$, then must be $u = b$ and is $x = ywwz = ybabbabz$, contradicting our assumption that x does not contain bb as subword.
Summarizing, if $x \in L$, then x contains at least one of the words aa, bb, $abab$ and $baba$ as subword.
Now it is easy to draw a FA that recognizes L:



2.58 Bekijk $L = \{x \in \{a,b\}^* : n_a(x) = n_b(x) \geq 0\}$ de taal bestaande uit alle woorden met evenveel a-en als b-en. Volgens exercise 5.13 en Example 5.7 is deze taal niet regulier. Nu geldt $L^* = L$, want elk concatenatie van (al dan niet verschillende) woorden uit L levert weer een woord met evenveel a-en als b-en. Dus L^* is niet regulier.

2.59 Bekijk *pal* de taal van alle palindromen over $\{a,b\}$. We weten dat *pal* niet regulier is, maar pal^* is wel regulier. Immers Λ is een palindroom en elke letter is een palindroom van lengte 1, dus elk woord over $\{a,b\}$ is een concatenatie van palindromen. Dus $pal^* = \{a,b\}^*$, een reguliere taal.

---

version 19 September 2011