Fundamentele Informatica II

Answer to selected exercises 3

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

M.M. Bonsangue (and J. Kleijn)

Fall 2011

3.1 a. $r = b^*(ab)^*a^*$: the word aab is not in the language, defined by r, since every a should be followed by a b or belong to a suffix of a's. Note that Λ, a, b, and all words of length 2 are in the language, defined by r. So, aab is of minimal length.
Another example is abb: every b should be preceded by a a unless it is part of a prefix of b's.
b. $r = (a^* + b^*)(a^* + b^*)(a^* + b^*)$: the words $abab$ and $baba$ are examples of words not belonging to the language defined by r, because r allows only a maximum of two a-b or b-a changes in a word when reading it from left to right. Verify that there are no shorter words (i.e. of length ≤ 3) not belonging to the language of r.
c. $r = a^*(baa^*)^*b^*$: the word bba does not belong to the language defined by r, because every occurrence of b should be followed by at least one a unles it belongs to a suffix of b's. Verify that all words of length ≤ 2 belong to the language.
d. $r = b^*(a + ba)^*b^*$: the word $abba$ does not belong to the language of r, because that requires that a b can only be followed by a b if it belongs to a prefix or suffix consisting of b's. Verify that all words of length ≤ 3 belong to the language.

3.3 a. $r(r^*r + r^*) + r^* = r^*$.

b. $(r + \Lambda)^* = r^*$.

c. The expression $(r + s)^*rs(r + s)^* + s^*r^*$ denotes all words that contains at least once rs (i.e. the expression $(r + s)^*rs(r + s)^*$) or do not contains any occurrence of rs at all (i.e. the expression s^*r^*). This is thus equivalent to $(r + s)^*$.

extra 1 The expression $(r(r + s)^*)^+$ can be simplified in $r(r + s)^*$

3.6 a. $(w)^*(z)^*$

b. $(w)^*a(w+z)^*$

c. $(w+z)^*(a+\Lambda)$

3.7 a. $b^*ab^*ab^*$

b. $(a+b)^*ab^*ab^*$

c. $\Lambda + b + (a+b)^*a + (a+b)^*bb$

e. $(b+ab)^*(\Lambda+a)$ and $(\Lambda+a)(b+ba)^*$

f. $b^*(ab^*ab^*)^*$

g. $(b+ab)^*(\Lambda+a+aa)(b+ba)^*$

h. $b^*(abbb^*)^*$

k. $(a+ba)^*b^*$.

l. $(a+b)^*(bab+aba)(a+b)^*$.

m. $((aa+ab(bb)^*ba) * (b+ab(bb)^*a)(a(bb)^*a) * (b+a(bb)^*ba))^*(aa+ab(bb)^*ba)^*(b+ab(bb)^*a)(a(bb)^*a)^*$.

m. $((aa+bb)^*(ab+ba)(bb+aa)^*(ba+ab))^*(aa+bb)^*(ab+ba)(bb+aa)^*$.

3.10 The reverse function `rev` assigns to each string its reversal (mirror image).
Formally, given an alphabet Σ, we define `rev`: $\Sigma^* \to \Sigma^*$ recursively by:
$\texttt{rev}(\Lambda) = \Lambda$ (no change)
$\texttt{rev}(xa) = a\texttt{rev}(x)$ for $x \in \Sigma^*$, $a \in \Sigma$ (last letter first, reverse the rest)
$\texttt{rev}(x)$ may be abbreviated as x^r.
For a language L we use L^r to denote the language consisting of the reversals of the words from L, thus $L^r = \{x^r \mid x \in L\}$.
a. Consider the regular expression $e = (aab + bbaba)^*baba$ defining the regular language $||e||$. Then the language $||e||^r$ can be defined by the regular expression $e_r = abab(baa + ababb)^*$; thus $||e_r|| = ||e||^r$.
b. In general we have the recursively defined function `rrev` which "reverses" regular expressions (in the sense that it yields a regular expression with a reversed semantics): $\texttt{rrev}(\emptyset) = \emptyset$; $\texttt{rrev}(\Lambda) = \Lambda$; $\texttt{rrev}(a) = a$ for all $a \in \Sigma$.
and for the composite elements:
if e_1 and e_2 are regular expressions, then $\texttt{rrev}(e_1+e_2) = \texttt{rrev}(e_1)+\texttt{rrev}(e_2)$;
$\texttt{rrev}(e_1e_2) = \texttt{rrev}(e_2)\texttt{rrev}(e_1)$; and $\texttt{rrev}(e_1^*) = (\texttt{rrev}(e_1))^*$.
Now we have to prove that this `rrev` has the property $||\texttt{rrev}(e)|| = ||e||^r$.
This is proved by induction on the structure of e:
$e = \emptyset$: then $||\texttt{rrev}(\emptyset)|| = ||\emptyset|| = ||\emptyset||^r$;
$e = \Lambda$: then $||\texttt{rrev}(\Lambda)|| = ||\Lambda|| = ||\Lambda||^r$;
$e = a$: then $||\texttt{rrev}(a)|| = ||a|| = ||a||^r$.
Induction step, assuming that $||\texttt{rrev}(e_1)|| = ||e_1||^r$ and $||\texttt{rrev}(e_2)|| = ||e_2||^r$:

$e = e_1 + e_2$: then
$||\mathtt{rrev}(e_1 + e_2)|| = ||\mathtt{rrev}(e_1)+\mathtt{rrev}(e_2)|| = ||\mathtt{rrev}(e_1)|| \cup ||\mathtt{rrev}(e_2)|| =$
(induction) $||(e_1)||^r \cup ||(e_2)||^r = (||e_1|| \cup ||e_2||)^r = ||e_1 + e_2||^r$;
$e = e_1e_2$: then
$||\mathtt{rrev}(e_1e_2)|| = ||\mathtt{rrev}(e_2)\mathtt{rrev}(e_1)|| = ||\mathtt{rrev}(e_2)|| \cdot ||\mathtt{rrev}(e_1)|| =$
(induction) $||(e_2)||^r||(e_1)||^r = (||(e_1)|| \cdot ||(e_2)|)^r = ||e_1e_2||^r$;
$e = e_1^*$: then
$||\mathtt{rrev}(e_1^*)|| = ||(\mathtt{rrev}(e_1))^*|| = ||\mathtt{rrev}(e_1)||^* =$
(induction) $(||e_1||^r)^* = (||e_1||^*)^r = ||e_1^*||^r$.
c. It follows from **b.** that the language L^r is regular whenever the language L is regular: we have seen that $L = ||e||$ implies that $L^r = ||\mathtt{rrev}(e)||$ and that $\mathtt{rrev}(e)$ is a regular expression follows immediately from the definition of `rrev` as given above.

3.18 See Figure 3.34.
a. We determine $\delta^*(1, aba)$. First observe that $\delta^*(1, \Lambda) = \Lambda(\{1\}) = \{1\}$; then $\delta^*(1, a) = \Lambda(\bigcup_{r\in\delta^*(1,\Lambda)} \delta(r, a) = \Lambda(\delta(1, a)) = \Lambda(\{2\}) = \{2, 3\}$. This means that processing symbol a from the initial state leads to state 2 or state 3.
We add b: $\delta^*(1, ab) = \Lambda(\delta(2, b) \cup \delta(3, b)) = \Lambda(\emptyset \cup \{3, 4\}) = \{3, 4, 5\}$ and so after ab we are in either state 3 or state 4 or state 5.
Finally we process another a: $\delta^*(1, aba) = \Lambda(\delta(3, a) \cup \delta(4, a) \cup \delta(5, a)) = \Lambda(\{4\} \cup \{4\} \cup \emptyset) = \Lambda(\{4\}) = \{4, 5\}$. Thus after reading aba we are in state 4 or in state 5 and since 5 is an accepting state, aba is accepted by M.
b. $abab$ is not accepted: from **a.** we know that $\delta^*(1, aba) = \{4, 5\}$. Thus $\delta^*(1, abab) = \Lambda(\delta(4, b) \cup \delta(5, b)) = \Lambda(\emptyset) = \emptyset$. Not only is there no accepting state for $abab$, it cannot even be completely processed!
c. $aaabbb$ is accepted by M (check!).

3.21: as in 3.18.

3.22
a. $\Lambda(\{2, 3\}) = \{2, 3, 5\}$.
b. $\Lambda(\{1\}) = \{1, 2, 5\}$.
d. To determine $\delta^*(1, ba) =$ first observe that $\delta^*(1, \Lambda) = \Lambda(\{1\}) = \{1, 2, 5\}$ (see above item b.). We thus have $\delta^*(1, b) = \Lambda(\bigcup_{p\in\delta^*(1,\Lambda)} \delta(p, b)) = \Lambda(\delta(1, b)\cup \delta(2, b) \cup \delta(5, b)) = \Lambda(\{6, 7\}) = \{1, 2, 5, 6, 7\}$. Finally we obtain $\delta^*(1, ba) = \Lambda(\bigcup_{p\in\delta^*(1,b)} \delta(p, a)) = \Lambda(\delta(1, a)\cup\delta(2, a)\cup\delta(5, a)\cup\delta(6, a)\cup\delta(7, a)) = \Lambda(\{3, 5\}) = \{3, 5\}$.

3.37 See Figure 3.36. Use the algorithm from the proof of Theorem 3.17

to obtain for each NFA an NFA without -Λ transitions accepting the same language.

a. In the table below, the first four columns describe the transition function of M. Its initial state is 1 and it has one final state: 4. As a useful extra we then compute $\Lambda(\{q\})$, the set consisting of all states that can be reached from q using only Λ-transitions. Next we compute the transitions of M', for each state q and symbol $c \in \{a, b\}$, using the formula $\delta'(q, c) = \delta^*(q, c) = \Lambda(\bigcup_{r \in \Lambda(q)} \delta(r, a))$.

Now we have the transitions of M' in the last two columns. Its initial state is again 1. As accepting states it has now 4 (just like M), but also the inital state 1, since $\Lambda(1) \cap \{4\} = \{4\} \neq \emptyset$!!!

q	$\delta(q, \Lambda)$	$\delta(q, a)$	$\delta(q, b)$	$\Lambda(\{q\})$	$\delta^*(q, a)$	$\delta^*(q, b)$
1	$\{4\}$	$\emptyset$	$\{2, 3\}$	$\{1, 4\}$	$\emptyset$	$\{2, 3, 5\}$
2	$\emptyset$	$\{3\}$	$\emptyset$	$\{2\}$	$\{3\}$	$\emptyset$
3	$\emptyset$	$\emptyset$	$\emptyset$	$\{3\}$	$\emptyset$	$\emptyset$
4	$\emptyset$	$\emptyset$	$\{5\}$	$\{4\}$	$\emptyset$	$\{5\}$
5	$\emptyset$	$\{4\}$	$\emptyset$	$\{5\}$	$\{4\}$	$\emptyset$

Draw M'.

3.41 a. We use the partial derivatives method to compute the transitions and the states of the *nondeterministic automaton* corresponding to the regular expression $E_0 = (b + bba)^*a$. First we note that $\Lambda \notin L(E_0)$, so E_0 is not an accepting state. We have $\partial_a((b+bba)^*a) = \partial_a((b+bba)^*)a \cup \partial_a(a) = \partial_a((b + bba))(b + bba)^*a \cup \{\Lambda\} = (\partial_a(b) \cup \partial_a(bba))(b + bba)^*a \cup \{\Lambda\} = \{\Lambda\}$. This is a new state, clearly accepting, that we denote by E_1. Continuing our calculation we obtain $\partial_b((b+bba)^*a) = \partial_b((b+bba)^*)a \cup \partial_b(a) = \partial_b((b+bba))(b+bba)^*a \cup \emptyset = (\partial_b(b) \cup \partial_b(bba))(b+bba)^*a = (\{\Lambda\} \cup \{ba\}(b+bba)^*a = \{(b+bba)^*a, ba(b+bba)^*a\}$. The first element in the set is E_0, and we denote the other one by E_2. Also here $\Lambda \notin L(E_2)$.

The transitions from $E_1 = \Lambda$ are calculated as follows: $\partial_a(\lambda) = \partial_b(\lambda) = \emptyset$.

The transitions from $E_2 = ba(b + bba)^*a$ are: $\partial_a(ba(b + bba)^*a) = \partial_a(b)a(b+bba)^*a = \emptyset$. and $\partial_b(ba(b+bba)^*a) = \partial_b(b)a(b+bba)^*a = \{\Lambda\}a(b+bba)^*a = \{a(b+bba)^*a\}$. The element of this set is a new state, say E_3, with $\Lambda \notin (E_3)$, thus non accepting.

The transitions from $E_3 = a(b+bba)^*a$ are: $\partial_a(a(b+bba)^*a) = \partial_a(a)(b+bba)^*a = \{\lambda\}(b + bba)^*a = \{(b + bba)^*a\}$. (this is just E_0) and $\partial_b(a(b + bba)^*a) = \partial_b(a)(b + bba)^*a = \emptyset a(b + bba)^*a = \emptyset$.

The resulting automaton (with E_0 as initial state) is summarized in the following table:

q	$\delta(q,a)$	$\delta(q,b)$	Accepting?
E_0	$\{E_1\}$	$\{E_0, E_2\}$	No
E_1	$\emptyset$	$\emptyset$	Yes
E_2	$\emptyset$	$\{E_3\}$	No
E_3	$\{E_0\}$	$\emptyset$	No

3.41 d. Let $E_0 = (a^*bb)^* + bb^*a^*$. We use the method of derivatives to find a *deterministic finite automaton* M accepting the language $L(E_0)$. Since $\Lambda \in L(E_0)$, the state E_0 is accepting. Further we calculate the two derivatives: $D_a((a^*bb)^* + bb^*a^*) = D_a((a^*bb)^*) + D_a(bb^*a^*) = D_a(a^*bb)(a^*bb)^* + D_a(b)b^*a^* = (D_a(a^*)bb + D_a(bb))(a^*bb)^* + \emptyset b^*a^* = (D_a(a)a^*bb + D_a(b)b)(a^*bb)^* + \emptyset = (a^*bb + \emptyset)(a^*bb)^* = (a^*bb)(a^*bb)^* = E_1$ and $D_b((a^*bb)^* + bb^*a^*) = D_b((a^*bb)^*) + D_b(bb^*a^*) = D_b(a^*bb)(a^*bb)^* + D_b(b)b^*a^* = (D_b(a^*)bb + D_b(bb))(a^*bb)^* + b^*a^* = (D_b(a)a^*bb + D_b(b)b)(a^*bb)^* + b^*a^* = (\emptyset + b)(a^*bb)^* + b^*a^* = b(a^*bb)^* + b^*a^* = E_2$ Note that Λ is in the language of E_2 but not in the languages of E_1. Thus only E_2 is accepting.

Next we calculate the derivatives of E_1. $D_a(a^*bb(a^*bb)^*) = D_a(a^*)bb(a^*bb)^* + D_a(bb(a^*bb)^*) = a^*bb(a^*bb)^* + \emptyset = a^*bb(a^*bb)^* = E_1$ and $D_b(a^*bb(a^*bb)^*) = D_b(a^*)bb(a^*bb)^* + D_b(bb(a^*bb)^*) = \emptyset + b(a^*bb)^* = b(a^*bb)^* = E_3$. Also E_3 is not accepting.

Next we calculate the derivatives of E_2. $D_a(b(a^*bb)^* + b^*a^*) = D_a(b(a^*bb)^*) + D_a(b^*a^*) = D_a(b)(a^*bb)^* + D_a(b^*)a^* + D_a(a^*) = \emptyset + \emptyset + a^* = a^* = E_4$ (an accepting state!). Further $D_b(b(a^*bb)^* + b^*a^*) = D_b(b(a^*bb)^*) + D_b(b^*a^*) = D_b(b)(a^*bb)^* + D_b(b^*)a^* + D_b(a^*) = (a^*bb)^* + b^*a^* + \emptyset = (a^*bb)^* + b^*a^* = E_5$ (again, an accepting state!).

The derivatives of E_3 are: $D_a(b(a^*bb)^*) = D_a(b)(a^*bb)^* = \emptyset = E_6$ and $D_b(b(a^*bb)^*) = D_b(b)(a^*bb)^* = (a^*bb)^* = E_7$ (an accepting state).

The derivatives of E_5 are $D_a((a^*bb)^* + b^*a^*) = D_a((a^*bb)^*) + D_a(b^*a^*) = D_a(a^*bb)(a^*bb)^* + D_a(b^*)a^* + D_a(a^*) = a^*bb(a^*bb)^* + \emptyset + a^* = a^*bb(a^*bb)^* + a^* = E_8$ (an accepting state) and $D_b((a^*bb)^* + b^*a^*) = D_b((a^*bb)^*) + D_b(b^*a^*) = D_b(a^*bb)(a^*bb)^* + D_b(b^*)a^* + D_b(a^*) = D_b(bb)(a^*bb)^* + b^*a^* + \emptyset = b(a^*bb)^* + b^*a^* = E_2$.

We skip the calculation of the derivatives of the other states, which are either easy or can be derived by the above calculations. The resulting automaton (with E_0 as initial state) is summarized in the following table:

q	RegExp	$\delta(q,a)$	$\delta(q,b)$	Accepting?
E_0	$(a^*bb)^* + bb^*a^*$	E_1	E_2	Yes
E_1	$(a^*bb)(a^*bb)^*$	E_1	E_3	No
E_2	$b(a^*bb)^* + b^*a^*$	E_4	E_5	Yes
E_3	$b(a^*bb)^*$	E_6	E_7	No
E_4	a^*	E_4	E_6	Yes
E_5	$(a^*bb)^* + b^*a^*$	E_8	E_2	Yes
E_6	$\emptyset$	E_6	E_6	No
E_7	$(a^*bb)^*$	E_1	E_3	Yes
E_8	$(a^*bb)(a^*bb)^* + a^*$	E_8	E_3	Yes

3.44 a. We add a new initial state $q_i \notin Q$ and a Λ-transition $\delta(q_i, \Lambda) = \{q_0\}$. All the rest remains unchanged.
b. We add a new single accepting state $q_f \notin Q$ and a Λ-transition $\delta(q, \Lambda) = \{q_f\}$ from every $q \in A$. All the rest remains unchanged.

3.46 The construction proposed does not work: take two automata $M_1 = (\{q_1\}, \{a,b\}, q_1, \{q_1\}, \delta_1)$ and $M_2 = (\{q_2\}, \{a,b\}, q_2, \{q_2\}, \delta_2)$, with $\delta_1(q_1, a) = \{q_1\}$, and $\delta_2(q_2, a) = \{q_2\}$. Then $L(M_1) = \{a^n | n \geq 0\}$ and $L(M_2) = \{b^n | n \geq 0\}$. In the new automata M_u we would have a path $\delta^*(q_1, aa\lambda)$ bringing to an accepting state (namely q_2), but $aab \notin L(M_1) \cup L(M_2)$.

3.51a See Figure 3.40 (a). We use the *algebraic* of Brzozowski to derive for the depicted automata a corresponding regular expression.
First we write the automaton in Figure 3.40 (a) as a system of 3 equations in three variables:

$$\begin{aligned} x_1 &= ax_3 + bx_2 \\ x_2 &= ax_1 + bx_3 \\ x_3 &= ax_2 + bx_1 + \Lambda \end{aligned}$$

By substituting x_3 in the first two equations we obtain the system

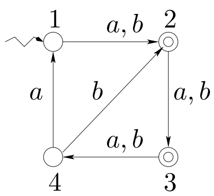
$$\begin{aligned} x_1 &= a(ax_2 + bx_1 + \Lambda) + bx_2 = abx_1 + (aa+b)x_2 + a \\ x_2 &= ax_1 + b(ax_2 + bx_1 + \Lambda) = bax_2 + ((a+bb)x_1 + b) \end{aligned}$$

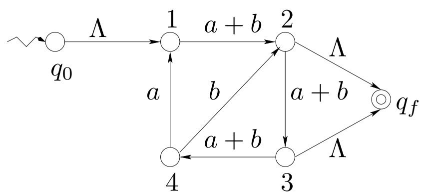
Using the Arden's lemma, we obtain that $x_2 = (ba)^*((a+bb)x_1 + b)$. If we substitute x_2 in the first equations we have

$$\begin{aligned} x_1 &= abx_1 + (aa+b)(ba)^*((a+bb)x_1 + b) + a \\ &= (ab + (aa+b)(ba)^*(a+bb))x_1 + ((aa+b)(ba)^*b + a) \end{aligned}$$

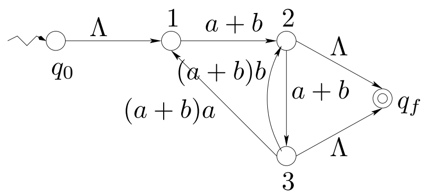
Using again Arden's lemma, we obtain that $x_1 = ((ab + (aa + b)(ba)^*(a + bb)))^*((aa + b)(ba)^*b + a)$. This is a regular expression denoting the same language of the automaton in Figure 3.40 (a).

3.51 c See Figure 3.40 (c). We use the *state removal method* of Brzozowski and McCluskey to derive for the depicted automata a corresponding regular expression.
First we add a new initial state q_0 without incoming transitions and a new (the only) final state q_f without outgoing transitions in such a way that the resulting NFA accepts the same language (see exercise 3.44). At the same time we combine with + the labels of parallel edges into a single regular expression.

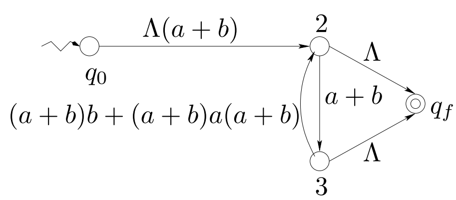




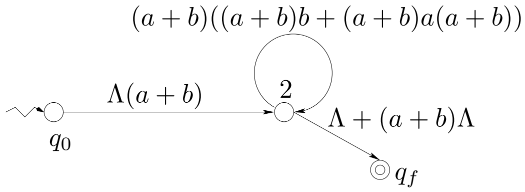
Now we remove state 4. Before deleting 4, we consider the transitions from 3 to 4 and from 4 to 1 and 2. This leads to the introduction of an arc labeled with $(a + b)a$ from 3 to 1 and an arc labeled with $(a + b)b$ from 3 to 2.



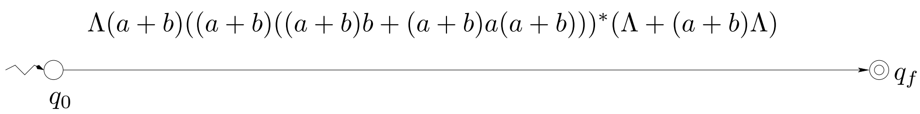
Then state 1 is removed. This leads to the introduction of an arc labeled with $\Lambda(a + b)$ from q_0 to 2 and an arc labeled with $(a + b)a(a + b)$ from 3 to 2. The latter is combined using + with the label $(a + b)b$ of the already existing arc from 3 to 2.



Then state 3 is removed. This leads to the introduction of an arc labeled with $(a+b)\Lambda$ from 2 to q_f which is combined with the existing parallel arc labeled with Λ. Also an arc from 2 to 2 is added which is labeled with $(a+b)((a+b)b+(a+b)a(a+b))$, a combination of the label of the arc from 2 to 3 and that of the arc from 3 to 2.



Finally, we remove state 2 and find a regular expression for $L(M)$.



$L(M)=\{a,b\}(\{a,b\}(\{a,b\}\{b\}\cup\{a,b\}\{a\}\{a,b\}))^*\{\Lambda,a,b\}.$

---

version 19 September 2011