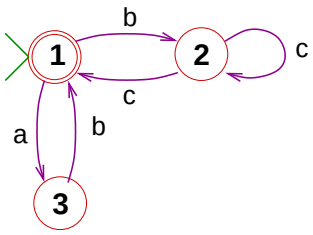
B501, Fall 2023

Assignment 5: Regular languages, other machines

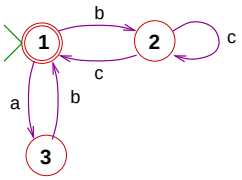
Solutions.

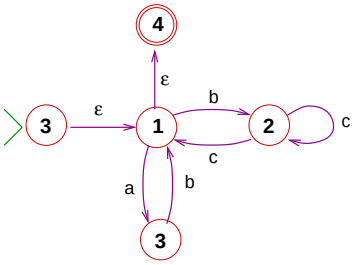
1. (30%) Convert each of the following NFA's N into an equivalent regular expression α (i.e. such that $\mathcal{L}(\alpha) = \mathcal{L}(N)$). Exhibit the stages of each conversion.

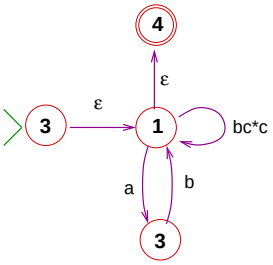
i.

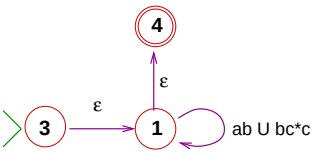


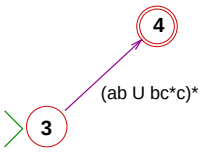
Solution.



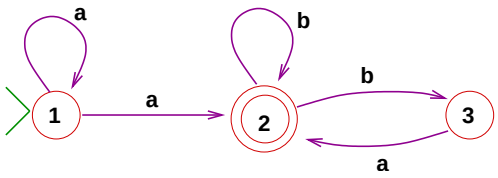




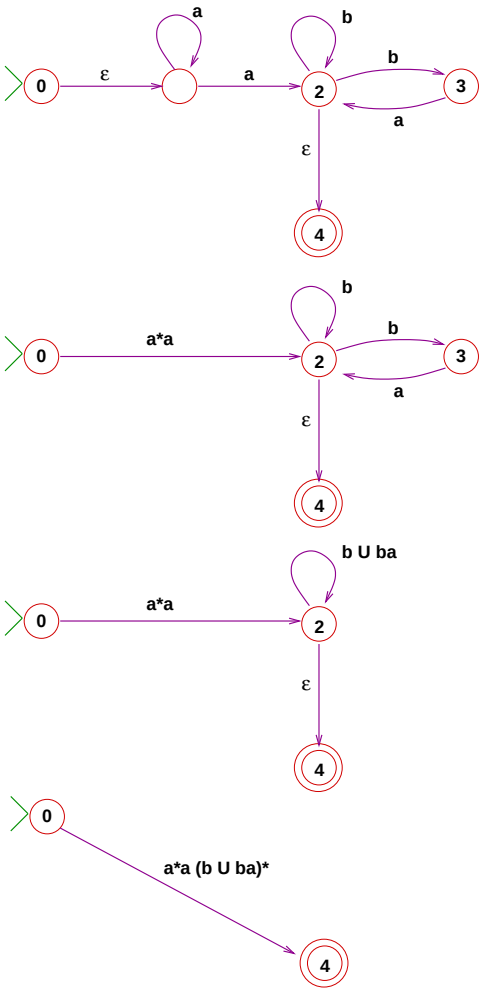




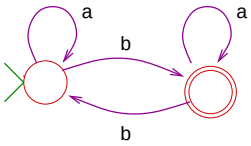
(a)



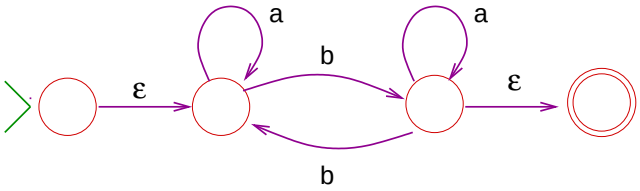
Solution.

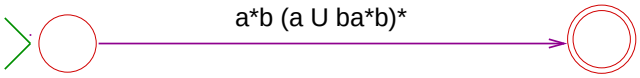


(b)



Solution.





2. (25%)

(a) Construct an LBA that recognizes the language

$$L = \{\mathtt{a}^n\mathtt{b}^m\mathtt{c}^{n+m} \mid n, m \geqslant 1\}.$$

We match a string of `a`'s and `b`'s with a string of `c`'s, deleting a terminal `c` for each initial `a` and `b`. Success requires that `a`'s precede `b`'s and `b`'s precede `c`'s. Here is the LBA in modular format, with S and Y as the start and accept states.

S	$\xrightarrow{\mathtt{>}\,(+)}$	L		C	$\xrightarrow{\mathtt{c}\,(+)}$	C	
L	$\xrightarrow{\mathtt{a}\,(\mathtt{>})}$	A		C	$\xrightarrow{\mathtt{u}\,(-)}$	R	
L	$\xrightarrow{\mathtt{b}\,(\mathtt{>})}$	B		R	$\xrightarrow{\mathtt{c}\,(\mathtt{u})}$	W	
A	$\xrightarrow{\sigma\,(+)}$	A	$(\sigma = \mathtt{>}, \mathtt{a})$	W	$\xrightarrow{\sigma\,(-)}$	W	$(\sigma \neq \mathtt{>})$
A	$\xrightarrow{\mathtt{b}\,(+)}$	B		W	$\xrightarrow{\mathtt{>}\,(+)}$	L	
B	$\xrightarrow{\mathtt{b}\,(+)}$	B		L	$\xrightarrow{\mathtt{u}\,(\mathtt{u})}$	Y	
B	$\xrightarrow{\mathtt{c}\,(+)}$	C					

(b) (25%) Give the computation-trace for **abcc**.

$$\begin{aligned}
(S, \underline{\mathtt{>}}\mathtt{abccu}) &\Rightarrow (L, \mathtt{>}\underline{\mathtt{a}}\mathtt{bccu}) \\
&\Rightarrow (A, \mathtt{>>}\underline{\mathtt{b}}\mathtt{ccu}) \\
&\Rightarrow (A, \mathtt{>>}\underline{\mathtt{b}}\mathtt{ccu}) \\
&\Rightarrow (B, \mathtt{>>b}\underline{\mathtt{c}}\mathtt{cu}) \\
&\Rightarrow (C, \mathtt{>>bc}\underline{\mathtt{c}}\mathtt{u}) \\
&\Rightarrow (C, \mathtt{>>bcc}\underline{\mathtt{u}}) \\
&\Rightarrow (R, \mathtt{>>bc}\underline{\mathtt{c}}\mathtt{u}) \\
&\Rightarrow (W, \mathtt{>>bc}\underline{\mathtt{u}}\mathtt{u}) \\
&\cdots \\
&\Rightarrow (W, \mathtt{>>}\underline{\mathtt{b}}\mathtt{cuu}) \\
&\Rightarrow (L, \mathtt{>>}\underline{\mathtt{b}}\mathtt{cuu}) \\
&\Rightarrow (B, \mathtt{>>>}\underline{\mathtt{c}}\mathtt{uu}) \\
&\Rightarrow (B, \mathtt{>>>}\underline{\mathtt{c}}\mathtt{uu}) \\
&\Rightarrow (C, \mathtt{>>>}\underline{\mathtt{c}}\mathtt{uu}) \\
&\Rightarrow (C, \mathtt{>>>c}\underline{\mathtt{u}}\mathtt{u}) \\
&\Rightarrow (R, \mathtt{>>>}\underline{\mathtt{c}}\mathtt{uu}) \\
&\Rightarrow (W, \mathtt{>>>}\underline{\mathtt{u}}\mathtt{uu}) \\
&\Rightarrow (W, \mathtt{>>>}\underline{\mathtt{u}}\mathtt{uu}) \\
&\Rightarrow (L, \mathtt{>>>}\underline{\mathtt{u}}\mathtt{uu}) \\
&\Rightarrow (Y, \mathtt{>>>}\underline{\mathtt{u}}\mathtt{uu})
\end{aligned}$$

assdsub Give the computation-trace for **abaccc**.

$$\begin{aligned}
(S, \underline{\mathtt{>}}\mathtt{abacccu}) &\Rightarrow (L, \mathtt{>}\underline{\mathtt{a}}\mathtt{bacccu}) \\
&\Rightarrow (A, \mathtt{>>}\underline{\mathtt{b}}\mathtt{acccu}) \\
&\Rightarrow (A, \mathtt{>>}\underline{\mathtt{b}}\mathtt{acccu}) \\
&\Rightarrow (B, \mathtt{>>b}\underline{\mathtt{a}}\mathtt{cccu})
\end{aligned}$$

The last configuration is terminal, because there is no transition for state B and symbol `a`.

3. (25%)

(a) Construct an LBA recognizing $L = \{w \cdot w^R \mid w \in \{\mathtt{a},\mathtt{b}\}^*\}$, where w^R is the reverse of w. Define your LBA in a modular format. [Hint: This is similar to the problem of accepting the strings $\mathtt{a}^n\mathtt{b}^n$ considered in class.]

Start state S, accept state Y.

$$
\begin{array}{rcll}
S & \xrightarrow{>\,(+)} & L & \\
L & \xrightarrow{\sigma\,(>)} & C_\sigma & \sigma = \mathtt{a},\mathtt{b}) \\
C_\sigma & \xrightarrow{\tau\,(+)} & C_\sigma & (\sigma = \mathtt{a},\mathtt{b},\ \tau \neq \sqcup) \\
C_\sigma & \xrightarrow{\sqcup\,(-)} & H_\sigma & (\sigma = \mathtt{a},\mathtt{b}) \\
H_\sigma & \xrightarrow{\sigma\,(\sqcup)} & R & (\sigma = \mathtt{a},\mathtt{b}) \\
R & \xrightarrow{\sigma\,(-)} & R & (\sigma = \mathtt{a},\mathtt{b}) \\
R & \xrightarrow{>\,(+)} & L & \\
L & \xrightarrow{\sqcup\,(\sqcup)} & Y &
\end{array}
$$

(b) Give the computation trace of your acceptor for **abba**.

$(S, \underline{>}\texttt{abba})$

$\Rightarrow (L, \texttt{>}\underline{\texttt{a}}\texttt{bba}) \Rightarrow (R, \texttt{>>bb}\underline{\sqcup}\,\sqcup) \Rightarrow (C_b, \texttt{>>>b}\underline{\sqcup}\,\sqcup)$

$\Rightarrow (C_a, \texttt{>>}\underline{\texttt{>}}\texttt{bba}) \Rightarrow (R, \texttt{>>b}\underline{\texttt{b}}\sqcup\,\sqcup) \Rightarrow (H_b, \texttt{>>>}\underline{\texttt{b}}\sqcup\,\sqcup)$

$\Rightarrow (C_a, \texttt{>>}\underline{\texttt{b}}\texttt{bba}) \Rightarrow (R, \texttt{>>}\underline{\texttt{b}}\texttt{b}\sqcup\,\sqcup) \Rightarrow (R, \texttt{>>>}\underline{\sqcup}\,\sqcup\,\sqcup)$

$\Rightarrow (C_a, \texttt{>>b}\underline{\texttt{b}}\texttt{a}) \Rightarrow (R, \texttt{>>}\underline{\texttt{>}}\texttt{bb}\sqcup\,\sqcup) \Rightarrow (R, \texttt{>>>}\underline{\texttt{>}}\sqcup\,\sqcup)$

$\Rightarrow (C_a, \texttt{>>bb}\underline{\texttt{a}}) \Rightarrow (L, \texttt{>>}\underline{\texttt{b}}\texttt{b}\sqcup\,\sqcup) \Rightarrow (L, \texttt{>>>}\underline{\sqcup}\,\sqcup\,\sqcup)$

$\Rightarrow (C_a, \texttt{>>bba}\underline{\sqcup}) \Rightarrow (C_b, \texttt{>>>}\underline{\texttt{>}}\texttt{b}\sqcup\,\sqcup) \Rightarrow (Y, \texttt{>>>}\underline{\sqcup}\,\sqcup\,\sqcup)$

$\Rightarrow (H_a, \texttt{>>bb}\underline{\texttt{a}}\sqcup) \Rightarrow (C_b, \texttt{>>>}\underline{\texttt{b}}\sqcup\,\sqcup)$

A. **i.** Construct a Turing transducer that replaces the last input symbol by `ba`.

Solution. Start state S, print state P.

$S \xrightarrow{\sigma\,(+)} S \quad (\sigma \neq \sqcup)$

$S \xrightarrow{\sqcup\,(\mathtt{a})} R$

$R \xrightarrow{\mathtt{a}\,(-)} B$

$B \xrightarrow{\sigma\,(\mathtt{b})} P$

ii. Give the computation trace of your transducer for input `baa`.

$(S, \geq \mathtt{baa}\sqcup) \Rightarrow (S, \mathtt{>}\underline{\mathtt{b}}\mathtt{aa}\sqcup)$
$\Rightarrow (S, \mathtt{>b}\underline{\mathtt{a}}\mathtt{a}\sqcup)$
$\Rightarrow (S, \mathtt{>ba}\underline{\mathtt{a}}\sqcup)$
$\Rightarrow (S, \mathtt{>baa}\underline{\sqcup})$
$\Rightarrow (R, \mathtt{>baa}\underline{\mathtt{a}})$
$\Rightarrow (B, \mathtt{>b}\underline{\mathtt{a}}\mathtt{a})$
$\Rightarrow (P, \mathtt{>b}\underline{\mathtt{b}}\mathtt{a})$

Start state S, print state P.

$$
\begin{array}{lll}
S & \xrightarrow{>(+)} & T \\
T & \xrightarrow{\sigma(+)} & D_\sigma \quad (\sigma \neq \sqcup) \\
D_\sigma & \xrightarrow{\tau(+)} & D_\tau \quad (\sigma, \tau \neq \sqcup) \\
D_\sigma & \xrightarrow{\sqcup(\sigma)} & R \quad (\sigma \neq \sqcup) \\
R & \xrightarrow{\sigma(-)} & R \quad (\sigma \neq >) \\
R & \xrightarrow{>(>)} & P
\end{array}
$$

4. (20%) Construct a Turing transducer over an alphabet Σ that swaps the first and last input symbols. For example, **abcde** is mapped to **ebcda**. (Single-letter strings and ε are mapped to themselves.)

$$
\begin{array}{lll}
S \xrightarrow{\sqcup\,(+)} C & & \\
C \xrightarrow{\sigma\,(+)} F_\sigma & (\sigma \in \Sigma) & \\
F_\sigma \xrightarrow{\tau\,(+)} F_\sigma & (\sigma, \tau \in \Sigma) & \\
F_\sigma \xrightarrow{\sqcup\,(-)} R_\sigma & (\sigma \in \Sigma) & \\
R_\sigma \xrightarrow{\tau\,(\sigma)} B_\tau & (\tau, \sigma \in \Sigma) & \\
B_\tau \xrightarrow{\sigma\,(-)} B_\tau & (\tau, \sigma \in \Sigma) & \\
B_\tau \xrightarrow{\sqcup\,(+)} L_\tau & (\tau \in \Sigma) & \\
L_\tau \xrightarrow{\sigma\,(\tau)} P & (\tau \in \Sigma) &
\end{array}
$$

The value of modular presentations of some Turing machines is illustrated by the following transition diagram for the modular program above already for the alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}\}$