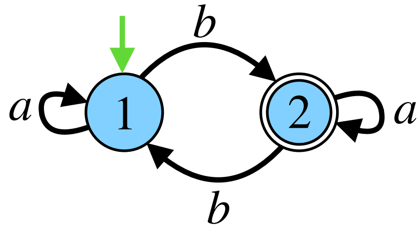
Nick G. Toth
Oct. 26, 2020
Ext: Oct. 27
CIS 420
Assignment III

Exercise 1 (Sipser 3e, Exercise 1.21)

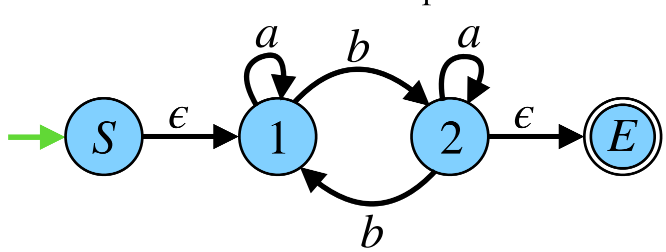
Note: Each step (for $k > 2$) of the CONVERT algorithm corresponds to the removal of a state. When I did this exercise on paper, I had numerous intermediate GNFAs where I only removed and combined edges. In copying over my solutions to this document, I have decided to show only the steps where I have removed a state, as the book does. I hope that's fine.

a)

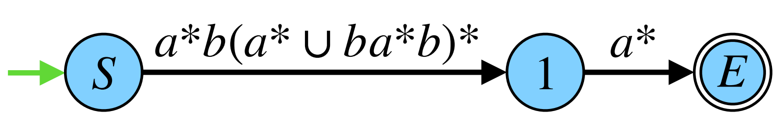
DFA



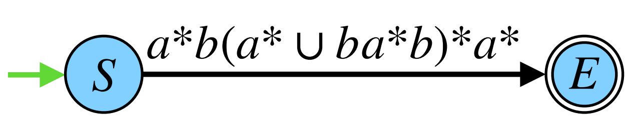
GNFA_1



GNFA_2



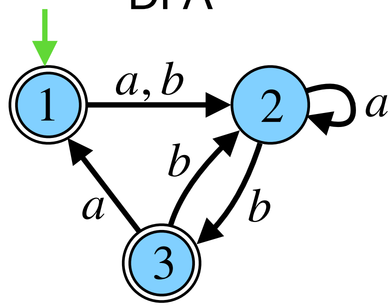
GNFA_3



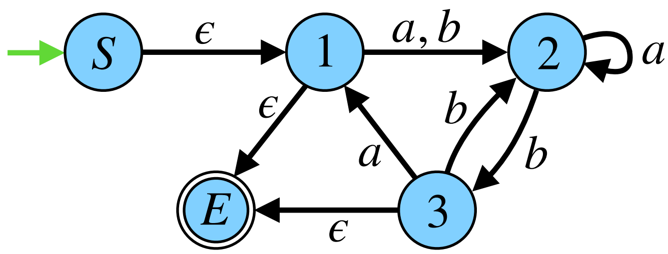
We can reduce the expression to $a^*b(a^* \cup ba^*b)^*$, since $a^* \subset (a^* \cup ba^*b)^*$.

$b)$

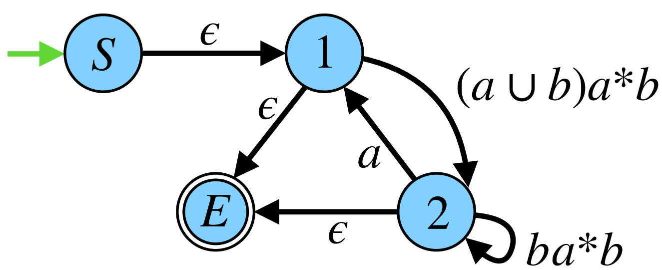
DFA



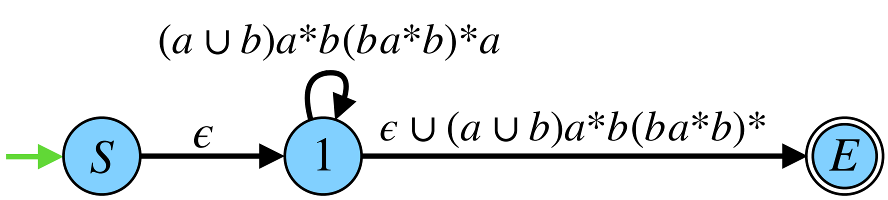
GNFA_1



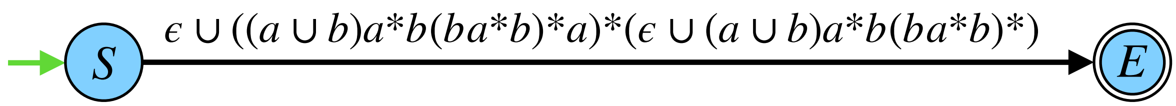
GNFA_2



GNFA_3



GNFA$_4$



So we have the expression $\epsilon \cup (\Sigma a^*b(ba^*b)^*a)^*(\epsilon \cup \Sigma a^*b(ba^*b)^*)$.

Reduced: $\epsilon \cup (\Sigma a^*b((b \cup a\Sigma)a^*b)^*(\epsilon \cup a))$.

Exercise 2 (Sipser 3e, Exercise 1.46 a, c)

$a)$ **Use the pumping lemma to prove that $\{0^n1^m0^n : m, n \geq 0\}$ is not regular.**

Proof:

Let A denote the language $\{0^n1^m0^n : m, n \geq 0\}$.
Now suppose A is regular, and let p be the pumping length of A.
By the pumping lemma, for any string $s \in A$, we have $s = xyz$ such that

$i)$ $\forall i \geq 0,\ xy^iz \in A$,
$ii)$ $|y| > 0$, and
$iii)$ $|xy| \leq p$.

Set $s = 0^p10^p$, and notice that $s \in A$ and $|s| = 2p + 1 > p$.
Now let $x = 0^i$, $y = 0^j$, and $z = 0^k10^n$ such that $i + j + k = p$ and $j > 0$.
Then $xy^0z = 0^i0^k10^p = 0^{i+k}10^p$.
But $i + k < p$, so this string cannot be an element of A.
Therefore A is not regular.
∎

$c)$ **Use the pumping lemma to prove that $\{w : w \in \{0,1\}^*$ is not a palindrome$\}$ is not regular.**

Proof:

Begin by noting that regular languages are closed under complement.
So $\{w : w \in \{0,1\}^*$ is not a palindrome$\}$ is regular iff $\{w : w \in \{0,1\}^*$ is a palindrome$\}$ is regular.

Now let A denote the language $\{w : w \in \{0,1\}^*$ is a palindrome$\}$.

Suppose A is regular, and let p be the pumping length of A.
By the pumping lemma, for any string $s \in A$, we have $s = xyz$ such that

$i)$ $\forall i \geq 0,\ xy^iz \in A$,
$ii)$ $|y| > 0$, and
$iii)$ $|xy| \leq p$.

Set $s = 0^p10^p$, and notice that $s \in A$ and $|s| = 2p + 1 > p$.
Now let $x = 0^i$, $y = 0^j$, and $z = 0^k10^n$ such that $i + j + k = p$ and $j > 0$.
Then $xy^0z = 0^i0^k10^p = 0^{i+k}10^p$.
But $i + k < p$, so this string cannot be an element of A.
So A is not regular, and therefore neither is $\{w : w \in \{0,1\}^* \text{ is not a palindrome}\}$.
∎

Exercise 3 (Sipser 3e, Exercise 1.53)

Note: In this exercise, I will use the symbol $\sim$, rather than $=$, to denote equality in the language, to avoid confusion.

Let $\Sigma = \{0,1,+,\sim\}$,
Add $= \{x \sim y + z : x, y, z$ are binary integers, and x is the sum of y and z$\}$.

Claim: Add is not a regular language.

Proof:
Now suppose Add is regular, and let p be the pumping length of Add.
By the pumping lemma, for any string $s \in$ Add, we have $s = xyz$ such that

$i)$ $\forall i \geq 0,\ xy^iz \in$ Add,
$ii)$ $|y| > 0$, and
$iii)$ $|xy| \leq p$.

Set $s = 1^p \sim 1^p + 0^p$, and notice that $s \in$ Add and $|s| > p$.
Now let $x = 1^i$, $y = 1^j$, and $z = 1^k \sim 1^p + 0^p$ such that $i + j + k = p$ and $j > 0$.
Then $xy^0z = 1^{i+k} \sim 1^p + 0^p$.
But this word is not an element of Add, since $1^{i+k} \neq 1^p$.
Therefore Add is not regular.
∎

Exercise 4 (Sipser 3e, Exercise 1.49)

$a)$ Let $B = \{1^k y : y \in \{0,1\}^*$ and y contains at least k 1's, for k ≥ 1 $\}$.
Show that B is a regular language.

Let $s = 1y$ for some string y containing at least one 1. Then $s \in B$, since no matter how many additional 1's appear in y, there will always be just a single 1 to the left of y.

It follows that B is recognized by the regular expression $1\Sigma * 1\Sigma *$.

$b)$ Let $C = \{1^k y : y \in \{0,1\}^*$ and y contains at most k 1's, for k ≥ 1 $\}$.
Show that B is not a regular language.

Proof:

Now suppose C is regular, and let p be the pumping length of C.
By the pumping lemma, for any string $s \in C$, we have $s = xyz$ such that

$i)$ $\forall i \geq 0,\ xy^i z \in C$,
$ii)$ $|y| > 0$, and
$iii)$ $|xy| \leq p$.

Set $s = 1^p 0 1^p$, and note that $s \in C$ and $|s| > p$.
Now let $x = 1^i$, $y = 1^j$, and $z = 1^k 0 1^p$ such that $i + j + k = p$.
Then $xy^0 z = 1^{i+k} 0 1^p$.
But $1^{i+k} 0 1^p$ is not an element of C because $p > i + k$.
Therefore C is not regular.
∎

Exercise 5 (Sipser 3e, Exercise 2.4 c,e)

I hope I've shown enough here. I found these exercises pretty intuitive, and I didn't feel like there was much to be said.

$c)$ Give a context-free grammar that generate the language $\{w : w$ has odd length$\}$.

$$S \to 0 \mid 1 \mid 00S \mid 01S \mid 10S \mid 11S.$$

e) Give a context-free grammar that generate the language $\{w : w = w^R\}$.

$$S \to \epsilon \,|\, 0 \,|\, 1 \,|\, 0S0 \,|\, 1S1.$$

Exercise 6 (Sipser 3e, Exercise 2.6 b)

Give a context-free grammar that generate the complement of $\{a^n b^n : n \geq 0\}$.

Begin by writing the given language as the union of $\{b^m a^n : m \neq n\}$ and the language generated by the regular expression $\Sigma^* b \Sigma^* a \Sigma^*$.

We can recognize the language $\{b^m a^n : m \neq n\}$ with the grammar

$S_1 \to A \,|\, B$

where A and B are designed to handle the cases $m > n$ and $m < n$, respectively. i.e.

$A \to aAb \,|\, aA \,|\, a$

$B \to aBb \,|\, Bb \,|\, b$

Now we convert the regular expression $\Sigma^* b \Sigma^* a \Sigma^*$ into the context free grammar

$S_2 \to CaCbC$

$C \to \epsilon \,|\, aC \,|\, bC$

Finally, we can combine S_1 and S_2 to obtain the solution

$S \to A \,|\, B \,|\, C$

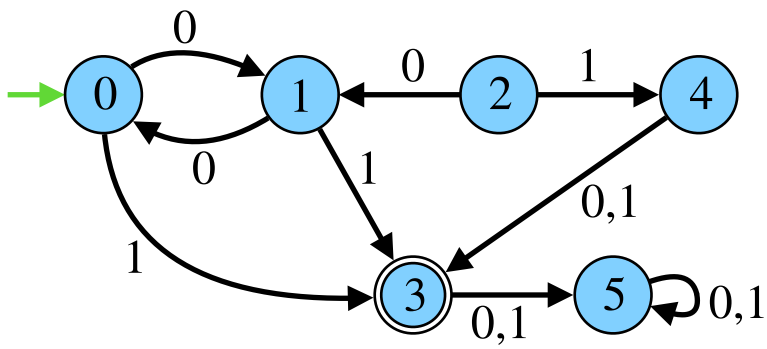
$A \to aAb \,|\, aA \,|\, a$

$B \to aBb \,|\, Bb \,|\, b$

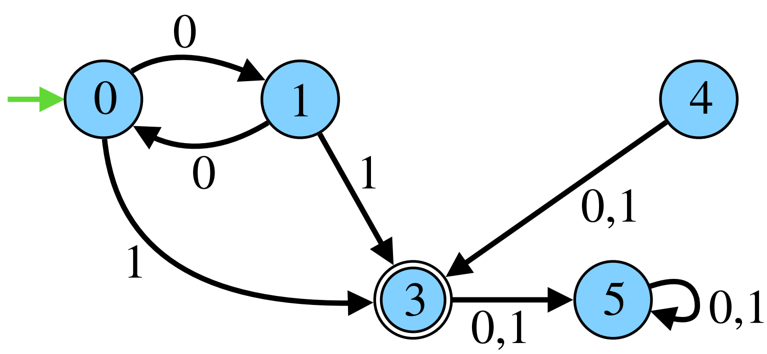
$C \to \epsilon \,|\, aC \,|\, bC$

Exercise 7 (Graduate Exercise - Purely Recreational)

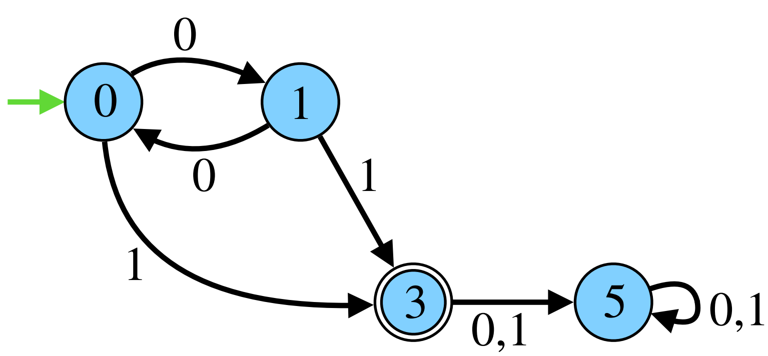
Apply the state minimization method to the following DFA.



Note: I did not follow any particular procedure. This solution was obtained intuitively. First observe that there are no arrows leading to state 2, so it can be removed.



Now state 4 can be removed for the same reason.



Finally, observe that states 0 and 1 generate an arbitrary string of zeroes. If a 1 is encountered at either state, then we end up at state 3, so we can replace state 1 by adding a 0-loop on state 0.