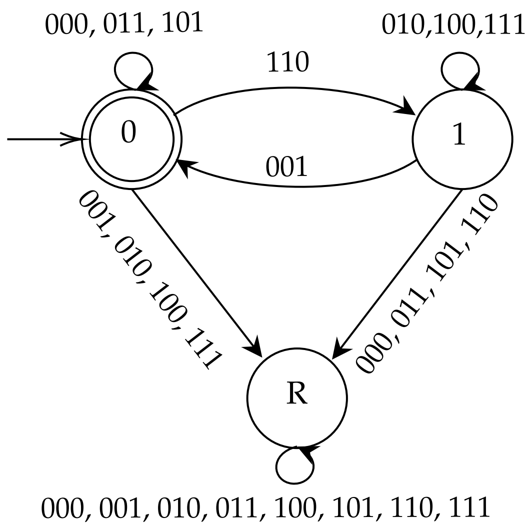
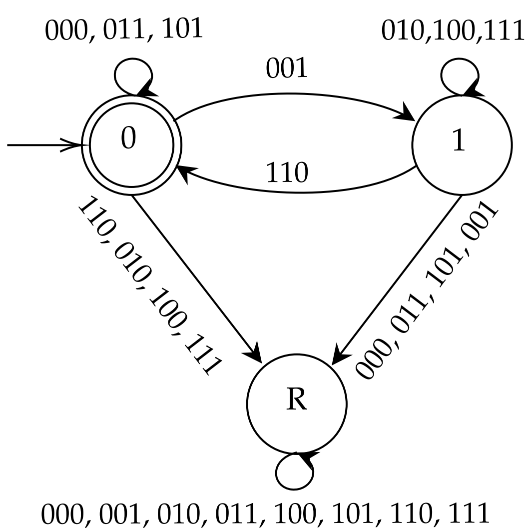
Name: Kevin Zhang

**Problem 1.**




(a) $A^R$ must be regular because it is accepted by the automaton above.

(b) $A$ must be regular language, because $A^R$ is a regular language. Since $A = (A^R)^R$, and the reversal operation is a regular operation, $A$ must be a regular language.




(c)

**Problem 2.**

(a) $baa \in a^*b^*a^*b^*$ is True, because the $^*$ operator is for zero or more of an element. $a^*b^*a^*b^*$ might look like $b^*a^*$, to which fits $baa$.

(b) $a^* \cup b^* = (a \cup b)^*$ is False, because the right hand side (inside parenthesis) is equivalent to $\{a, b\}$. This, combined with the star operator, means any string with any combination of $a$s or $b$s will work. In contrast, the left hand side accepts only either strings with only $a$s, or strings with only $b$s, and the empty string. $aba$ will be accepted by the right, but not by the left.

(c) $(a^*b^*)^* = (a \cup b)^*$ is True. The left side can form any string with any combination of $a$s or $b$s. This is possible, because the inside of the parenthesis expands out to be $\{\varepsilon, a, b, ab, b, ...\}$, which has $\{\varepsilon, a, b\}$ as a subset.

(d) $b^*a^* \cap a^*b^* = a^* \cup b^*$ is True, because the only intersection the left side can have is the same letter repeated. $baaaa$ cannot be in $b^*a^* \cap a^*b^*$, because there is no way to construct it using $a^*b^*$. Similar argument can be made about $aaaaab$. Therefore, the only elements in such an interesection are either $a^*$ or $b^*$, the union of which is $a^* \cup b^*$.

(e) $(ab)^*a = a(ba)^*$ is True. Both sequences form a palindrome-like chain of $aba$. The left side looks like $\{a, aba, ababa, abababa, ...\}$, and the right side forms the same sequence.

(f) $a^*b^* \cap c^*d^* = \varnothing$ is False. $\varepsilon$ is in both sets, so the intersection should be $\{\varepsilon\}$.

(g) $abcd \in (a(cd)^*b)^*$ is False. The right side has $(cd)^*$ between $a$ and $b$, and in $abcd$, $cd$ is after $ab$.

**Problem 3.**

(a) $(b^*(\varepsilon \cup a \cup ab^*a \cup ab^*ab^*a)b^*)$

(b) $\varepsilon \cup ((a \cup b)^*a) \cup ((a \cup b)^*bb)$

(c) $(b^*ab^*ab^*ab^*)^*$

**Problem 4.**

(a) $(a \cup b)^*$ because that represents any length string of $a$s and $b$s. $(a \cup \varepsilon)b^*$ is not a strong restriction, because both parts can be $\varepsilon$, and any $a$ or $b$s can then get folded into $(a \cup b)^*$.

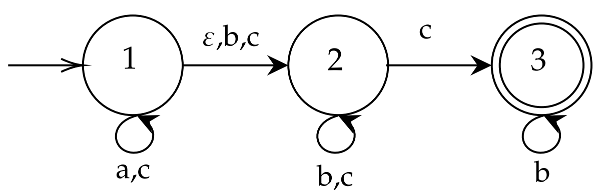
(b) $(a \cup b)^*$ because all the other elements are subsets of $(a \cup b)^*$.

(c) $a^*b$ because if we start looking at the accepted elements, they are $\{b, ab, aab, ...\}$.

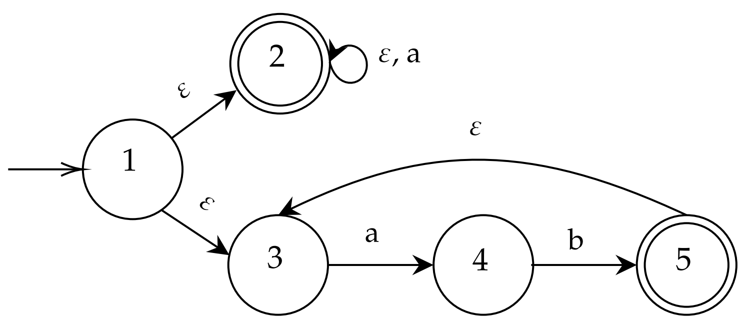
(d) $(a \cup b)^*$ because the left side is all elements that end in $b$ (plus $\varepsilon$), and the right side is all elements that end in $a$ (plus $\varepsilon$). Since all substrings of $\{a, b\}$ end in $a$ or $b$, or is $\varepsilon$, $(a \cup b)^*$ is my answer.

(e) $(a \cup b)^* a (a \cup b)^*$ because the expression cannot be simplified further. The expression is all strings that have at least one $a$ in it, and there is no other way to indicate all possible strings of $\{a, b\}$ BEFORE and AFTER.

**Problem 5.**




**Problem 6.**




**Problem 7.**

(a) The primary differences between NFA and DFA are as follows

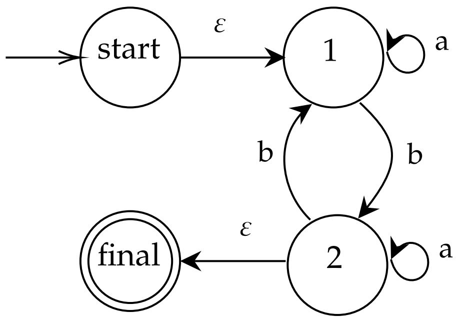
1. There are multiple possible out arrows for a single value
2. $\varepsilon$ is allowed on an arrow
3. Some values lead to $\varnothing$ / they don't lead anywhere.

(b) The primary difference between GNFA and NFA is that GNFA uses regular expressions on arrows, whereas NFA does not.

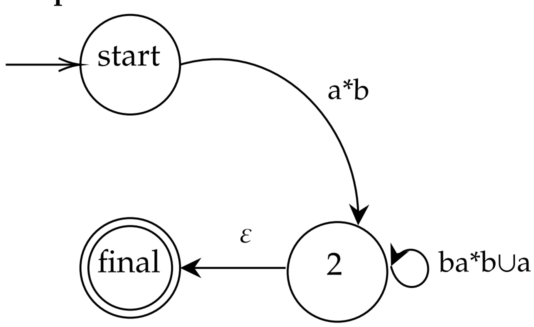
**Problem 8.**

(a)

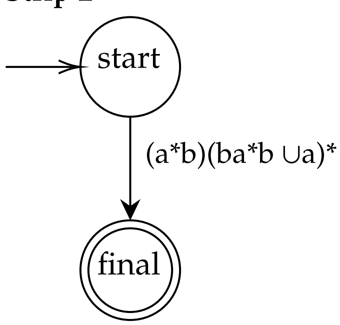
1. **NFA to GNFA**



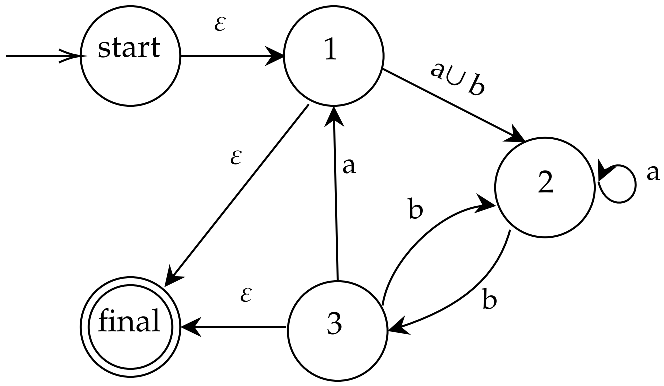

2. **Strip 1**



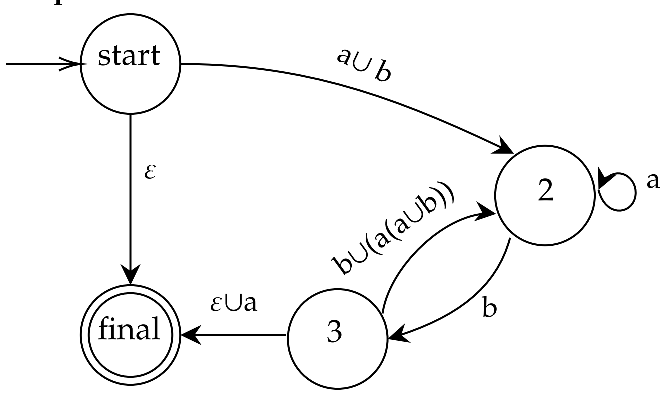

3. **Strip 2**



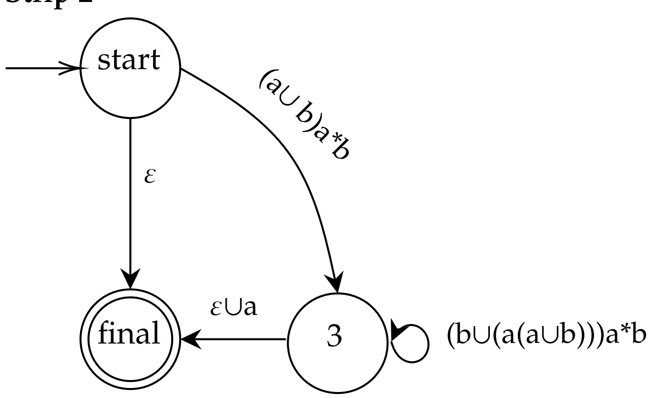
(b) 1. **NFA to GNFA**



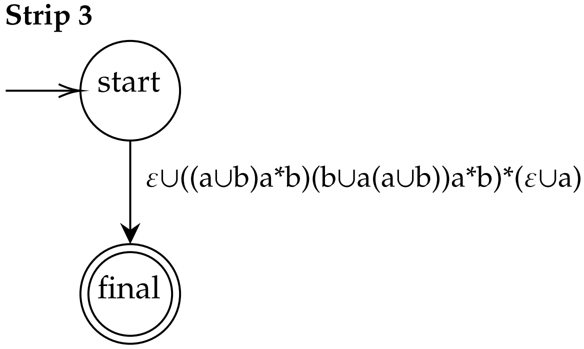

2. **Strip 1**




3. **Strip 2**




4. **Strip 3**



**Problem 9.**

(a) $p \neq 0$ because the only zero-length string in the language is $\varepsilon$, which cannot be pumped.
$p = 1$ works, because we can pump the first character in the string, whether it is 0, which will be covered by $0^*$, or it is 1, which is covered by $1^*$. There are no strings in the language that is 1 followed by a 0, so $0^*1^*$ covers all our cases.

(b) $p \neq 1$ because we can simply choose one character in the 01, and then we will have either $0^*1$ or $01^*$, neither of which fits $(01)^*$.
$p = 2$ works, because we can then pick the first occurence of 01 and pump that.

(c) $p \neq 0$ because the only zero-length string in the language is $\varepsilon$, which cannot be pumped.
$p = 1$ works, because the set of words of at least length is $\emptyset$, to which the statment becomes vacuously true.

(d) $p \neq 2$ because 00 cannot be pumped. The restriction is that there are exactly two zeroes, which pumping will break if we pick either 0.
$p = 3$ works, because we can pump the 1, wherever it appears.

(e) $p \neq 3$ because in 100, 10 cannot be pumped down, only up.
$p = 4$ works, because the first $w \in A$ we can use is 10100, in which we pick the second 10. This can be pumped up or down, since $(10)*$ will fit under $(11 * 0)*$.

(f) $p \neq 4$ because 1011 is of length 4, and anything we pump will not be in the set.
$p = 5$ works, because there are not strings of at least length 5, which leaves us $\emptyset$ to pump, which is vacuously true.

(g) $p \neq 0$ because the only zero-length string in the language is $\varepsilon$, which cannot be pumped.
$p = 1$ works, because we can simply pump the first character. $\Sigma^*$ is a language that includes all substrings with 1 and 0.

**Problem 10.**

$p \neq 3$ because a counter example is 000, where no matter what is pumped, there will eventually be 0000 as a substring.
$p = 4$ works, because we can pump the first four letters in $w$. We know that the word does not contain 0000 nor 1111, and and repeating the same word cannot produce 0000 or 1111, because there is no 4-letter word that both begins and ends with 00 and is not 0000. Same is true for 1111.

**Problem 11.**

(a) Assume that $L = \{ww^R | w \in \{a, b\}^*\}$ is regular. Then, there must be some pumping length $p$. Consider $w = a^p b^p b^p a^p$. We have $w \in L$ and $|w| \geq p$. If we let $y = a^p$, and $x = \varepsilon$ and $z = b^p b^p a^p$, we can pump $y$ up or down, which will break the symmetry of $a$'s to the left and right of the center. This broken symmetry creates a contradiction, because $L = \{ww^R | w \in \{a, b\}^*\}$ is symmetric words only. Thus, $L$ cannot be regular.

(b) Assume that $L = \{w\bar{w} | w \in \{a, b\}^*\}$ is regular. Then, there must be some pumping length $p$. Consider $w = a^p b^p$. We have $w \in L$ and $|w| \geq p$. If we let $y = a^p$, $x = \varepsilon$, and $z = b^p$, we can pump $y$ up or down, breaking the equal parity of $a$ and $b$s. This generates a word that is not in $L$, which creates a contradiction. Thus, $L$ cannot be regular.

**Problem 12.**

Assume that $L = \{a^t | t \text{ is a prime number}\}$ is regular. Then, there must be some pumping length $p$. Consider $w = a^p a^{t-p}$ where $t$ is a prime number and $t \geq p$. Then, $w \in L$ and $|w| \geq p$. If we let $y = a^p a^{t-p}$, $x = \varepsilon$ and $z = \varepsilon$. If we then pump $y$, to some whole number $n$, we would get $w = a^{np} a^{n(t-p)}$, which can be simplified to $w = a^{nt}$. This $w \notin L$, because $nt$ is by definition not prime. Therefore, we have a contradiction. $L$ cannot be regular.