1. Here is the diagram:
2. Say that some language $L$ is accepted by an FSA $F$. Let $\{a_1, ..., a_n\}$ be the alphabet. Call the complementary language $L^*$. We need to design a machine that will accept $L^*$. The basic idea is simple: There are two ways that a string can fail to be in $L$: either $F$ reads the whole string but terminates in a non-final state, or it terminates before reading the entire string. Thus we construct our machine $F^*$ to accept $L^*$ by changing all the final states of $F$ to non-final states of $F^*$ and non-final states of $F^*$ to final states of $F$. That will take care of the first issue.

To take care of the second issue, we’ll add a new state $q_{terminate}$, which just reads everything it sees and stays put. We add the tuples $(q_{terminate}, a_i, q_{terminate})$ for every $i$.

Make this a final state of $F^*$. When does the machine go into the state $q_{terminate}$? Here’s when:

*Terminating Instruction:* For any state $q$ of $F$, say there is a letter $a_i$ for which $q$ has no instructions. That is: there is no state $q'$ such that $(q, a_i, q')$. Then we add the tuple $(q, a_i, q_{terminate})$.

Note that every transition of $F$ is a transition of $F^*$; the only new transitions in $F^*$ are those added by the “terminating instruction”.

The diagram is a bit cluttered with all the letters labelling arrows, so if you find it easier to read, here is a color coded version (Gold = [Chemical: au] = a, blue = b, cyan = c)
Claim: $F^*$ accepts a string if and only if the string is not accepted by $F$.

Proof (Claim): There are two directions we need to take care of.

$\Rightarrow$

Say that $\sigma = \sigma_1 \cdots \sigma_k$ is a string that is accepted by $F$. Then $F$ reads $\sigma$ and terminates in some final state $\bar{q}$. Since every transition of $F$ is a transition of $F^*$, this means that $F^*$ reads $\sigma$ and terminates in $\bar{q}$. But by the design of $F^*$, since $\bar{q}$ is a final state of $F$, it is not a final state of $F^*$. That is, $F^*$ does not accept $\sigma$.

[Bit of elementary logic: Proving "If not-A then not-B", as we have done here, also proves "If A then B". Hence we have proven the $\Rightarrow$ direction.]

$\Leftarrow$

Say that $\sigma = \sigma_1 \cdots \sigma_k$ is a string that is not accepted by $F$. Then one of two things happens when $F$ is fed $\sigma$: either i) $F$ doesn’t read the whole string, or ii) $F$ reads the whole string and terminates in a non-final state $\bar{q}$. If ii), then since the transitions of $F$ are also transitions of $F^*$, this means that $F^*$ reads all of $\sigma$ and halts in the same state $\bar{q}$, which by construction of $F^*$ is a final state of $F^*$. So $F^*$ accepts $\sigma$. If i), then on reading some letter $\sigma'_j$ in $\sigma$ $F$ doesn’t read further. Say $F$ is in state $\bar{q}$ when it is reading $\sigma_j$. Since $F$ reads no further, there must be no transition $\langle \bar{q}, \sigma_j, q' \rangle$ for any $q'$ in $F$. Thus by the construction of $F^*$ $F^*$ has a transition $\langle \bar{q}, \sigma_j, q_{\text{terminate}} \rangle$; in $q_{\text{terminate}}$, the rest of the string $\sigma$ is read. Since $q_{\text{terminate}}$ is a final state of $F^*$, $F^*$ accepts $\sigma$ in this case too.

3. a) False. The language containing all the concatenations of $a$ and $b$ is regular, since it is the Kleene $*$ of $\{a\} \cup \{b\}$. The set DUPLICATES described on p. 109 - 110 and in lecture is a subset of $\langle \{a\} \cup \{b\} \rangle^*$ but it is not regular. Similarly, the language $L = \{aa\cdots a bb\cdots b / n \in \mathbb{N}\} = \{\epsilon, ab, aabb, aaabbb, aaaaabbb, \ldots\}$ that we proved not to be regular on p.12 of this week’s notes is a subset of $\langle \{a\} \cup \{b\} \rangle^*$.

b) True. We showed in question 2 that the complement of a regular language is regular. Note that $\{xy/x \in L \text{ and } y \not\in L\} = \{x|x \in L\} \cup \{y|y \not\in L\}$