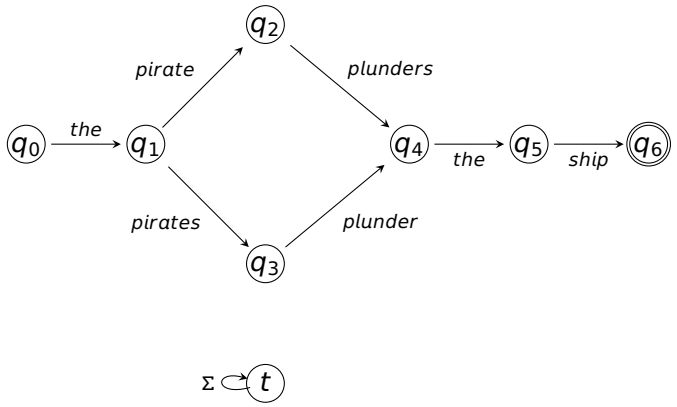
**Student's note:** In the process of typesetting this assignment, some DFA diagrams became incredibly convoluted and unreadable. For this reason, I've made the convention that the initial node will always be $q_0$, and whenever there is a node missing a path, it should be mapped to the trash node $t$.

1. Given all the recent hype about ChatGPT, you have decided to start doing work in natural language processing. But before trying to make a high-powered LLM, you've decided to see what you can do with some simpler tools: regular languages and NFAs. Consider the following input alphabet: {the, salty, pirate, pirates, plunder, plunders, ship}.

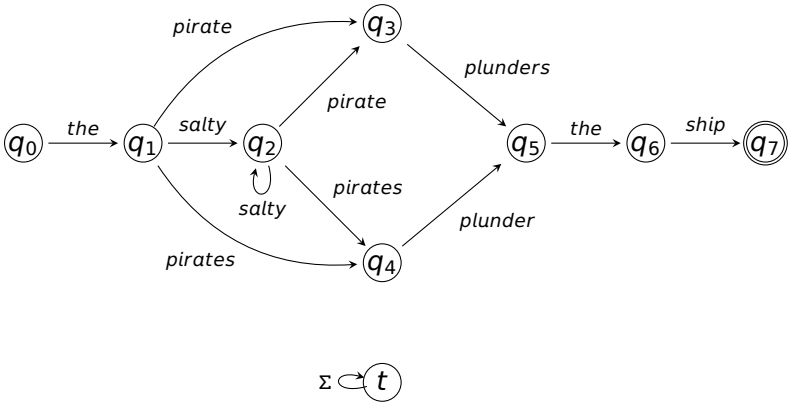
   (a) Construct a DFA which accepts the following language:

   **Solution:** Please refer to student's note at the top of the first page.



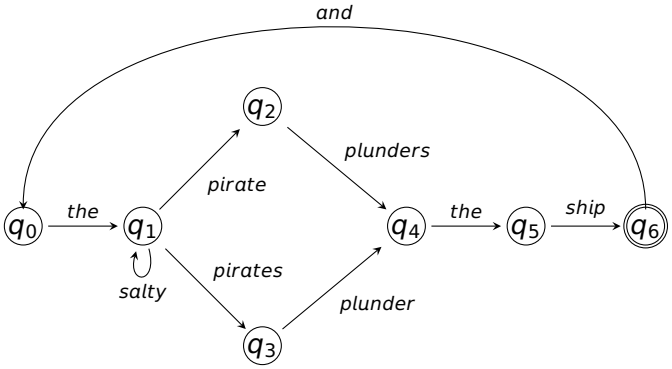

   (b) Construct a DFA accepting the language {the pirate plunders the ship, the pirates plunder the ship, the salty pirate plunders the ship, the salty pirates plunder the ship, the salty salty pirate plunders the ship, the salty salty pirates plunder the ship,. . . }

   **Solution:** Please refer to student's note at the top of the first page.



(c) Construct an NFA accepting the sentences accepted by the automaton in (b) plus those formed by conjoining such sentences with and, e.g. the salty pirate plunders the ship and the salty salty pirates plunder the ship and the pirate plunders the ship

**Solution:** Please refer to student's note at the top of the first page.




(d) Construct regular expressions for the automata in (a), (b), (c).

**Solution:**

(a) the (pirate plunders + pirates plunder) the ship
(b) the (salty)* (pirate plunders + pirates plunder) the ship
(c) the (salty)* (pirate plunders + pirates plunder) the ship (and the (salty)* (pirate plunders + pirates plunder) the ship)*

(e) Suppose our alphabet is {the, pirate, pirates, admiral, admirals, sailor, sailors, plunder, plunders, pursue, pursues, hate, hates, ship}
Consider sentences like the following:

the pirate plunders the ship
the pirate the admiral pursues plunders the ship
the pirate the admiral the sailor hates pursues plunders the ship (i.e. the sailor hates the admiral who pursues the pirate who plunders the ship)
the pirate plunders the ship
the pirates the admirals hunt plunders the ship
the pirates the admirals the sailors hate hunt plunders the ship
the admirals the sailors pursue

i.e. sentences of the form (the + [Noun])$^n$ + [Transitive Verb]$^n$ + the ship
Is the language that accepts these sentence regular? Why or why not?

**Solution:** The regular expression: "(the [Noun])$^n$ [Transitive Verb]$^n$ the ship" generates the language, and since the language is generated by a regular expression it must itself be regular.

2. Prove that $L(r^{**}) = L(r^*)$

**Proof:** Let $s \in L(r^{**})$. Then $s$ is some repetition of a string of the form $r^*$, $s = (r^*)^k$ for $k \geq 0$, which can be again rewritten $s = (r^l)^k$ for nonnegative $l$, and if we concatenate a string matching $r$ $l$ times, and then concatenate that $k$ times, we will be concatenating some string $kl$ times. So $s = r^{kl}$, a nonnegative repetition of $r$. So $s \in L(R^*)$.
Then let $s \in L(r^*)$. Then $s = r^k$ for some $k \geq 0$. But $k = 1k$, so write $s = r^{1k} = (r^1)^k$. So $s \in L(r^{**})$.
And so we have shown $L(r^*) \subseteq L(r^{**})$, and $L(r^*) \supseteq L(r^{**})$, so $L(r^*) = L(r^{**})$ ■

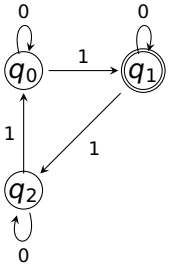
3. Show how, given any NFA, you can construct an equivalent one which has no transition arrows into the original state

   **Solution:** Let $N = (Q, \Sigma, \delta, q_0, A)$ be a nondeterministic finite automata. We follow the typical power set conversion to a DFA. Let $D = (Q', \Sigma', \delta', q_0', A)$ be given by: $q_0' = \{q_0\}^\lambda$. Then, we construct $Q'$ as per usual, adding subsets of $Q$ to $Q'$ and adjusting the maps of $\delta'$ accordingly. However, in the case that we are about to map something to $q_0'$ (i.e. there exists some $q' \in Q', a \in \Sigma$ so that $\delta'(q', a)$ would be $q_0'$), we must instead create a new $p \in Q'$, and have $\delta'(q', a) = p$. Then for any other element that would map to $q_0'$, we also map it to $p'$. Finally we adjust $p$ to have all the characteristics of $q_0'$, having all outgoing maps identical ($\delta'(p', a) = \delta'(q_0', a) \forall a \in \Sigma$), and setting $p'$ to an accepting state exactly if $q_0'$ is an accepting state.

4. Construct DFAs for the following regular languages over $\{0, 1\}$. Use as few states a possible.

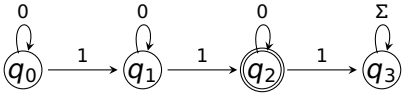
   (a) The set of all strings containing a total of $n$ 1's where $n \equiv 1 \pmod 3$.

   **Solution:**



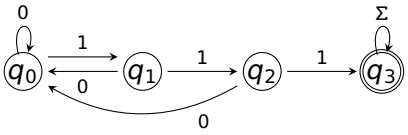

   (b) The set of all strings containing exactly two 1's.

   **Solution:**




   (c) The set of all strings containing a block of at least three consecutive ones:

   **Solution:**



(d) 1(010*1)(0|1)*

**Solution:**