31. We'll prove that the perfect shuffle of two regular languages A and B is regular by constructing a DFA that recognizes it. Let's say that A and B are recognized respectively by the DFAs $D_A = (Q_A, \Sigma, \delta_A, q_A, F_A)$ and $D_B = (Q_B, \Sigma, \delta_B, q_B, F_B)$. We construct a DFA D_S as follows.

$$\begin{aligned}
Q_S &= Q_A \times Q_B \times \{A, B\} \\
\Sigma_S &= \Sigma \\
\delta_S((q, q', A), c) &\to (\delta_A(q, c), q', B) \\
\delta_S((q, q', B), c) &\to (q, \delta_B(q', c), A) \\
q_S &= (q_A, q_B, A) \\
F_S &= F_A \times F_B \times \{A\}
\end{aligned}$$

D_S tracks the progress of the input string through both D_A and D_B by keeping track of the states of these DFAs, and alternating between their transition function upon reading each symbol. D_S starts out tracking the start states of both DFAs, and expecting to effect the next transition on D_A. D_S reaches a final state upon reading the input string iff the input string has a form $a_1b_1a_2b_2 \cdots a_kb_k$ where $a_1a_2 \cdots a_k \in L(D_A)$ and $b_1b_2 \cdots b_k \in L(D_B)$, because such and only such strings can take the state of D_S to some state in $F_A \times F_B \times \{A\}$.

This concludes the proof.

33. We'll prove that $DROPOUT(A)$ is a regular language by constructing an NFA that recognizes it. Since A is a regular language, some DFA $D = (Q, \Sigma, \delta, q_0, F)$ recognizes it. We construct NFA N as follows.

$$\begin{aligned}
Q_N &= Q \cup copy(Q) \\
\Sigma_N &= \Sigma \\
\delta_N(q, c) &\to \delta(q, c) \\
\delta_N(q, \epsilon) &\to copy(q) \\
\delta_N(copy(q), c) &\to copy(\delta(q, c)) \\
q_{0N} &= q_0 \\
F_N &= copy(F)
\end{aligned}$$

For every state q in Q, we'll create a copy state $copy(q)$. Call the set of all copy states $copy(Q)$. We'll augment the transition function as follows. Firstly, we'll add identical transitions within $copy(Q)$ to what existing in Q. Secondly, we'll add ϵ transitions from Q to $copy(Q)$. The latter makes skipping a symbol equivalent to transitioning from some state q to $copy(q)$. The idea is that once the NFA has skipped a symbol, it enters some state in $copy(Q)$ and stays within $copy(Q)$ till the very end, since there is no way to go back into Q from $copy(Q)$. And further that unless the NFA skips a symbol (through an ϵ transition from Q to $copy(Q)$), it won't enter a state in $copy(Q)$ and thus won't be able to accept the input because $F_N = copy(F) \subseteq copy(Q)$. Therefore, N recognizes $DROPOUT(A)$.

This concludes the proof.

42. We will construct a DFA such that when it has read a certain prefix of the string, the DFA is in a state that represents the remainder of that prefix modulo n. For any integer, there are n possible remainders modulo n. Correspondingly, we create n separate states and label them $0, 1, \cdots n-1$. Naturally, the only accept state is state 0. Assume we have read some prefix of the input and are at state i. If we read bit b next, where $b \in \{0, 1\}$, then, the resulting integer is $(2i+b)\%n$ modulo n. Accordingly, we'd have two possible state transitions, one from state i to state $(2i)\%n$ upon bit 0, and from state i to state $(2i+1)\%n$ upon bit 1. That completes the construction of the DFA, and thus the proof.

43. We need to prove two things, firstly that for each regular language, we can construct an all-NFA that recognizes it, and secondly that any all-NFA recognizes a regular language. The first part is clear because for each regular language, we can construct a DFA, and a DFA is also an all-NFA. We prove the second part below by constructing a DFA equivalent to any given all-NFA.

Given an all-NFA $(Q, \Sigma, \delta, q_0, F)$, construct an equivalent DFA D as follows. We'll assume that F is not empty, otherwise, it is trivial to construct an equivalent DFA, one that doesn't accept anything.

$$\begin{aligned} Q_D &= P(Q) \\ \Sigma_D &= \Sigma \\ \delta_D((\{q_{a_1}, q_{a_2}, \cdots q_{a_k}\}), c) &\to (\cup_{i=1}^{k} eps(\delta(q_{a_i}, c))) \\ q_{0D} &= (eps(q_0)) \\ F_D &= P(F) \setminus \{\} \end{aligned}$$

This construction is very similar to the one used to prove that NFAs are equivalent to DFAs. We create a separate state in the DFA to represent each subset of states of the all-NFA (to represent the idea that the all-NFA could be in any subset of its states at a given point of time in its execution). The transition function of the DFA builds on the transition function of the all-NFA in such a manner that when the all-NFA transits from some subset of states to another subset of states upon a certain symbol, the DFA transits from the state that represents the first subset to the state that represents the ϵ closure of the second subset. The DFA starts out the state that represents the ϵ closure of the start state. And finally, by making sure that the states of the DFA that correspond to the subsets of the powerset of the all-NFA's final states (except the empty set) are the final states of the DFA, we ensure that the DFA accepts a string if an only if the all-NFA accepts it.

This concludes the proof.

44. We have to demonstrate a language $A_k \subseteq \{0,1\}^*$ for which there is a k state DFA, but for which there cannot be a $k-1$ state DFA. Consider the language $A_k = \{w^r | w \in \{0,1\} \wedge (r \equiv k-1 \mod k)\}$, i.e. the language of strings whose length is $k-1 \mod k$. It is easy to see that the following k state DFA recognizes it.

$$
\begin{aligned}
Q &= \{0, 1, \cdots k-1\} \\
\Sigma &= \{0, 1\} \\
\delta(q, 0) &= (q+1)\%k \\
\delta(q, 1) &= (q+1)\%k \\
q_0 &= 0 \\
F &= k-1
\end{aligned}
$$

Now, let us suppose that there were some $k-1$ state DFA that also recognizes this language. Obviously, the DFA must accept a $k-1$ length string, because a $k-1$ length string belongs in the above language A_k. Further, the DFA must visit exactly k states while operating on this string. Since the DFA has only $k-1$ states, two of these k states must be the same (due to the pigeonhole principle). If we snip off the substring that was observed by the DFA between two visits to this state, we'd be left with a string of length smaller than $k-1$ but that is also accepted by this DFA. This is a contradiction because a string of length smaller than $k-1$ cannot be $\equiv k-1 \mod k$ and therefore does not belong to the above language A_k. Therefore, it must be that the above language A_k cannot be recognized by a DFA with $k-1$ states (or fewer for that matter). This concludes the proof.