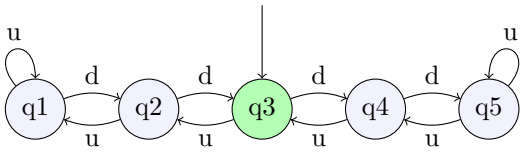
Selected solutions week 1 - IN2080

The following are solutions to some of the weekly tasks which we discuss in the group sessions.

1.3



1.31

Let M and N be two arbitrary DFAs, (Q_M, Σ_M, δ_M, $q1_M$, F_M) and (Q_N, Σ_N, δ_N, $q1_N$, F_N) respectively. The idea of this proof is to jump between the two machines every time a symbol is read. We do this by constructing a new DFA M' = $(Q, \Sigma, \delta, q1, F)$ such that

- $Q = Q_M \times Q_N \times \{M, N\}$

  States are tuples of a state from M, a state from N, and either M or N. Having states from both machines in each state of the new, allows us to know which state the previous machine we simulated is currently in. The last element of the tuples states which machine we are to simulate next.

- $\Sigma = \Sigma_M \cup \Sigma_N$

- $\delta((q_i, q_j, x), a) = \begin{cases} (\delta_M(q_i, a), q_j, N) & \text{if } x = M \\ (q_i, \delta_N(q_j, a), M) & \text{if } x = N \end{cases}$

  The state we are in tells us which machine to simulate, i.e. which transition function to use. Notice we also switch which machine to simulate next.

- $q1 = (q1_M, q1_N, M)$

  The initial state is the state composed of the initial states of both machines, and M. The reason it is the state with M and not N, is that the

words of the *perfect shuffle* of two languages A and B **must** start with a symbol from a word of A.

- $F = \{(q_i, q_j, M) | q_i \in F_M \text{ and } q_j \in F_N\}$

  The final states are the states composed of the final states of both machines, and M. Here M is chosen as well, as the words in the *perfect shuffle* must end with a symbol from a word of the second language. If a state contains M, it must mean that machine N was last simulated (or we have not simulated anything yet).

This DFA is able to know the current position of both machines at the same time. It is also capable of knowing which machine to simulate at any given time, and the transition function is built such that we'll always simulate the right one. Since M and N were arbitrarily chosen, we are able to make this machine for any two DFAs, and thus construct a DFA for the *perfect shuffle*. As such, the class of regular languages is closed under *perfect shuffle*.

1.32

To show that the class of the regular languages is closed under *shuffe*, we can use the same construction as in the previous exercise, although slightly modified. The only difference is with the transition function, where we introduce some ϵ-transitions, and thus constructing an NFA instead of a DFA:

- $\delta'((q_i, q_j, x), a) = \{\delta((q_i, q_j, x), a)\}$ if $a \neq \epsilon$
- $\delta'((q_i, q_j, M), \epsilon) = \{(q_i, q_j, N)\}$
- $\delta'((q_i, q_j, N), \epsilon) = \{(q_i, q_j, M)\}$

The difference here, aside from the obvious change of the result being a single state to a set of a single state, is that we've added ϵ-transitions from a state (q_i, q_j, M) to (q_i, q_j, N), and vice versa. These transitions allow the machine to non-deterministically guess when to switch, by being able to "return" to simulating the same machine as it was in the previous step. Note that the words in a *shuffle* of two languages are not required to begin or end with a part of a word from a particular language, i.e. a word can begin with a part of a word from the second language, and end with a part of a word from the first. Thus, the ϵ-transitions won't enable the machine to recognize words it shouldn't.