Prob. 1	Prob. 2

Team members: Küng, Pirelli, Schubert, Dousse, Vu

Problem 1.

As a DFA only accepts Σ^* if all reacheable states from q_0 are accepting ones, we can make a breadth (or depth) first search on these states, while remembering the one visited, to avoid loops and know when to stop.

If ever a non accepting state is found, reject. Else accept.

Therefore, as a depth or breadth first search is in Polynomial time, and the rest is in constant time, our whole algorithm runs in Polynomial time.

Therefore $ALL_{DFA} \in P$

Problem 2.

Let L be a language in P. Let A be a polynomial-time algorithm that decides L. We want to create A', a polynomial type algorithm that decides L^*, given an input w consisting of characters $c_1c_2...c_n$.
To do so, using dynamic programming, we build a matrix M such that $M[i,j]$ is true iff $c_i...c_j \in L^*$. This matrix is built with a bottom-up approach, where $M[i,j]$ can be true if $c_i...c_j \in L$, or if $c_i...c_j$ can be divided in segments $s \in L \forall s$ (and therefore $c_i...c_j \in L^*$).
The process of building that matrix is done in polynomial time, more precisely in $O(a^3)$ where a is the complexity of A since, for each pair $(i,j) \in (1..n, 1..n)$, there is one execution of A to check if $c_i...c_j \in L$ and $(j-i)$ executions of A to check if $c_i...c_j$ is made up of two parts $p \in L \forall p$.
Once this matrix is built, $w \in L^* \; iff \; M[1,n]$ is true.