Lecture 10 Addendum
Converting NFAs to DFAs

This algorithm takes an NFA as input and produces a DFA accepting the same language. The technique used involves constructing the DFA state by state, with each state in the new machine corresponding to an group of states from the NFA which are equivalent with regards to reachability.
In order to state the algorithm, we must first make the following definition:

> Given a state s in an NFA, the **ϵ-closure** of s is the smallest set of states satisfying the following conditions:
>
> 1. $s \in \epsilon\text{-closure}(s)$
> 2. If the state t is in the ϵ-closure(s) and there is an ϵ-transition from t to another state u, then u is also in the ϵ-closure(s).
>
> That is, the ϵ-closure of s is all the states that can be reached from s using only ϵ-transitions.

By extension, we also define the ϵ-closure of a set of states S as

$$\epsilon\text{-closure}(S) = \bigcup_{s \in S} \epsilon\text{-closure}(s)$$

In the following algorithm we will denote states in the original NFA by subscripting them with an N and those in the DFA under construction by subscripting them with a D. Recall that each state in the DFA corresponds to a set of states from the NFA. The algorithm proceeds as follows:

1. If S_{0N} is the start state of the NFA, let S_{0D}, the start state of the DFA, be $S_{0D} = \epsilon\text{-closure}(S_{0N})$. Initially, the state S_{0D} is *unmarked*.
2. Select an unmarked state S_{iD} in the DFA under construction and mark it. If there are no unmarked states, go to step 5.
3. For each symbol a in the input symbol set of the NFA:
   (a) Let $T = \{t_N | \text{there is a state } s_N \in S_{iD} \text{ and a transition in the NFA from } s_N \text{ to } t_N \text{ on input } a\}$.
   (b) Let $S_{jD} = \epsilon\text{-closure}(T)$.
   (c) If S_{jD} is not already a state in the DFA, add it unmarked.
   (d) Add a transition from S_{iD} to S_{jD} on input a.
4. Return to step 2.
5. The final (accepting) states of the DFA are exactly those which contain a final state of the NFA.