SE 3310b

Theoretical Foundations of Software Engineering

Week 3:

Equivalence of DFAs and NFAs. Closure Properties of Regular Languages.

Aleksander Essex

Western Engineering

Non-determinism Continued

Equivalence of NFAs and DFAs

A natural question to ask is how DFAs and NFAs are related. On first glance it seems like NFAs can do more, but is this really the case? The first step to answering this question is to ask: Given a DFA does there exist an equivalent NFA? By equivalent, we mean that they accept the *same* language.

Theorem 1 (All DFAs have an equivalent NFA)**.**

Every deterministic finite automaton has an equivalent non-deterministic finite automaton.

Equivalence of NFAs and DFAs

Proof. The proof is by definition: every DFA is also an NFA. It just so happens that DFAs do not make use of the extra ”power” of NFAs (i.e., multiple transitions and ϵ-transitions).

Equivalence of NFAs and DFAs

Now comes the harder part: prove that every NFA has an equivalent DFA (or not). If every NFA has an equivalent DFA, then NFAs ultimately have no more computational ”power” than DFAs.

Theorem 2 (All NFAs have an equivalent DFA)**.**

Every non-deterministic finite automaton has an equivalent deterministic finite automaton.

Equivalence of NFAs and DFAs

Proof. To show every NFA has an equivalent DFA, it would be sufficient to prove the existence of a general algorithm to covert any NFA into an equivalent DFA.

Converting NFAs into DFAs

We begin by reviewing each component of the definition of a DFA to see what needs to be ”adjusted.”

1. **States.** The first thing to consider is the finite set of states Q. In an NFA, the computation can be in a combination of states at the same time. In a DFA it can only be in one state at any time. IDEA: let sets of states in an NFA each be a single state in the corresponding DFA.

Converting NFAs into DFAs

1. **States Cont'd.** Example: suppose an NFA transitions to states q_0 and q_1. In our corresponding DFA we can define an "amalgamated" or "unified" state "q_0 and q_1". How many such states do we need? If our NFA has $|Q_N| = k$ states, our DFA has $|Q_D| = \mathcal{P}(Q_N) = 2^{|Q_N|}$. Example: $Q_N = \{q_0, q_1, q_2\}$ then $Q_D = \{\phi, \{q_0\}, \{q_1\}, \{q_2\}, \{q_0, q_1\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}\}$ where ϕ is the "dead" state (which we will omit for notational clarity).

   This is the complete set of states that our DFA *could* possibly use. Of course as we fill in the transition table we may find some (or even most) states go unused.

Converting NFAs into DFAs

2. **Alphabet.** The alphabet stays the same: $\Sigma_N = \Sigma_D$
3. **Starting state.** Let $q_0 \in Q_N$ be the starting state of our NFA. The starting state of the corresponding DFA is $\{q_0\} \cup \delta(q_0, \epsilon)$, i.e., combination of q_0 and any other states reachable from q_0 by an ϵ-transition.
4. **Final states.** Let F_N be the set of final states of our NFA. Recall the NFA accepts if any of the computational paths accepts. Taking this over to a DFA means designating any of the "amalgamated" states as accept if they contain some $q_i \in F_N$. Example: if $F_N = \{q_1\}$ then $F_D = \{\{q_1\}, \{q_1, q_0\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}\}$

Converting NFAs into DFAs

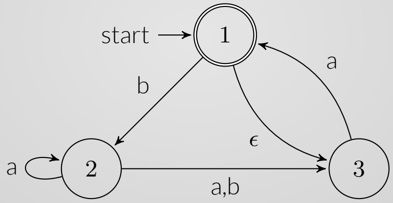
5. **Transition Function.** Filling in the transition function δ_D proceeds in two parts. **Part 1:** we write a table with single states for rows and letters for columns. We use the NFA's transition function δ_N to fill in this table converting sets of states to "amalgamated" states. But we still have to account for ϵ-transitions. The way we do this is as follows: if we transition from a state q_i to a state q_j, and q_j contains an ϵ-transition to state q_k , then in effect q_i transitions to $\{q_j\} \cup \{q_k\}$. This continues recursively until there are no new ϵ-transitions to follow.

Converting NFAs into DFAs

5. **Transition Function Cont'd. Part 2**: Now that we have the transitions for the single states written, we add new rows to the table corresponding to the amalgamated states i.e., $\{\{q_0, q_1\}, \{q_0, q_2\}\}$, etc.). We fill in these cells by taking the union of the cells of the corresponding single states. E.g. to fill in the cell $\delta_D(\{q_0, q_1\}, a)$ we take $\delta_D(\{q_0\}, a) \cup \delta_D(\{q_1\}, a)$

Example 1

Convert the following NFA to an equivalent DFA:



Example 1

1. **States.** Q_D will be end up being some subset of the powerset of Q_N, i.e., $Q_D \subset \mathcal{P}(Q_N)$.
2. **Alphabet**. $\Sigma_D = \Sigma_N = \{a, b\}$
3. **Start state**. The NFA's start state is 1, but there's an ϵ-transition to 3. The DFA's start state, therefore, is the amalgamated state $\{1, 3\}$.
4. **Final states**. $F_N = \{1\}$, so F_D is the set of all amalgamated states that include 1, i.e., $F_D = \{\{1\}, \{1, 2\}, \{1, 3\}, \{1, 2, 3\}\}$

Example 1

5. **Transition function.** In part 1 of the conversion we start out by writing out the transitions for single the single states taking in to account the ϵ-transitions:

	a	b
$\{1\}$		$\{2\}$
$\{2\}$	$\{2, 3\}$	$\{3\}$
$\{3\}$	$\{1, 3\}$	

Example 1

5. **Transition function Cont'd.** Now we fill in the rest of the amalgamated states by taking the unions of the single states:

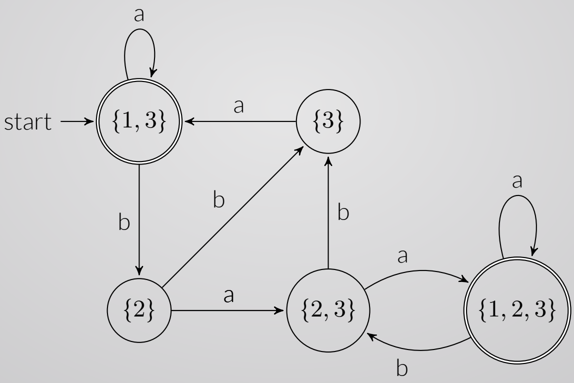
	a	b
$\{1\}$		$\{2\}$
$\{2\}$	$\{2,3\}$	$\{3\}$
$\{3\}$	$\{1,3\}$	
$\{1,2\}$	$\{2,3\}$	$\{2,3\}$
$\rightarrow \{1,3\}$	$\{1,3\}$	$\{2\}$
$\{2,3\}$	$\{1,2,3\}$	$\{3\}$
$\{1,2,3\}$	$\{1,2,3\}$	$\{2,3\}$

Example 1

Minimization. We’re using more states than we need. If you look down the columns you’ll notice no current state/symbol combination will lead to states $\{1\}$ or $\{1, 2\}$. Since neither of those states are a start state, we can never reach them, so we can eliminate them.

Example 1

Finally, putting it all together, the resulting DFA looks like this:



Equivalence of DFAs and NFAs

What does all this mean? Well recall we proved (by definition) that all DFAs have an equivalent NFA. Then, we demonstrated an algorithm showing all NFAs can be converted in to an equivalent DFA.

Corollary: NFAs have no more computational ”power” than DFAs. In fact the set of languages recognized by DFAs and NFAs are **equivalent**, i.e., the regular languages.