G52MAL
Machines and their Languages
Lecture 5: Equivalence between NFAs and DFAs

Thorsten Altenkirch
based on slides by Neil Sculthorpe

Room A10
School of Computer Science
University of Nottingham
United Kingdom
txa@cs.nott.ac.uk

9th February 2012
DFAs are NFAs

- A DFA is just a special case of an NFA, where there is:
  - exactly one initial state;
  - exactly one transition from each state per symbol.
- Thus all DFA transition diagrams also define an NFA that accepts the same language.
Given a DFA

\[ A = (Q, \Sigma, \delta_A, q_0, F) \]

an equivalent NFA \( N(A) \) can be constructed as follows:

\[ N (A) = (Q, \Sigma, \delta_{N(A)}, \{ q_0 \}, F) \]

where

\[ \delta_{N(A)} (q, x) = \{ \delta_A (q, x) \} \]
NFA Observations

- An NFA is always in a set of states.
- When reading an input symbol, the machine enters a new set of states.
- How many possible sets of states are there?
  - For each state, the machine is either in that state or not — i.e. 2 possibilities per state.
  - Thus $2^{|Q|}$ possibilities.
- This could be a lot, but it is finite.
- So we could convert an NFA to a DFA by taking each set of NFA states to be a single DFA state!
Converting NFAs to DFAs: The Subset Construction

Given an NFA

\[ A = (Q, \Sigma, \delta_A, S, F_A) \]

an equivalent DFA \( D(A) \) can be constructed as follows:

\[ D(A) = (\mathcal{P}(Q), \Sigma, \delta_{D(A)}, S, F_{D(A)}) \]

where

\[ \delta_{D(A)}(P, x) = \bigcup \{ \delta_A(q, x) \mid q \in P \} \]

\[ F_{D(A)} = \{ P \mid P \in \mathcal{P}(Q) \land (P \cap F_A \neq \emptyset) \} \]
Summary

A DFA is a special case of an NFA.

An NFA can be converted to a DFA using the Subset Construction.

Thus DFAs and NFAs are interconvertible, and therefore equivalent in the sense that they characterise exactly the same class of languages: the Regular Languages.
Recommended Reading

- Introduction to Automata Theory, Languages, and Computation (3rd edition), pages 60–71
- G52MAL Lecture Notes, pages 11–13