Formal Language & Finite Automata

Non-deterministic finite state automata

Dr Noleen Koehler

University of Leeds

Reading

All reading is from the "Introduction to the theory of computation" - Micheal Sipser

- Notation - Pages 1 - 21 (recommended)
- Deterministic finite state automata - Pages 31 - 47 (essential)
- Non-deterministic finite state automata - Pages 47 - 56 (essential)

Regular language

Definition (Regular Language)
A language A is **Regular** if there exists a deterministic finite state automaton M such that $L(M) = A$.

Here $L(M)$ denotes the set of stings which the finite state automaton M accept, also called the language that M defines.

Non-deterministic finite state automata

We now introduce a new model of computation.

Definition (Non-deterministic finite automaton)
A **non-deterministic finite automaton** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the **states**,
2. Σ is a finite set called the **alphabet**,
3. $\delta : Q \times \Sigma_\epsilon \to \mathcal{P}(Q)$ is the **transition function**,
4. $q_0 \in Q$ is the **start state**, and
5. $F \subseteq Q$ is the set of **accepting states**.

Where $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$.

Non-deterministic finite state automata - Computation

Let $N = (Q, \Sigma, \delta, q_0, F)$ be a non-deterministic finite automaton and w be a string over the alphabet Σ. We say that N accepts w if we can write $w = y_1, \ldots, y_n$ where $y_i \in \Sigma_\epsilon$ for $1 \leq i \leq n$ and there exists a sequence of states $r_0, \ldots, r_n$ where $r_i \in Q$ where $0 \leq i \leq n$, such that;

1. $r_0 = q_0$,
2. $r_{i+1} \in \delta(r_i, w_{i+1})$, for $0 \leq i \leq n-1$, and
3. $r_n \in F$.

Non-deterministic finite state automata

Theorem (Equivalence)
Every non-deterministic finite automaton has an equivalent deterministic finite automaton.

Equivalence - Proof

Let $N = (Q, \Sigma, \delta, q_0, F)$ be a non-deterministic finite automaton that recognises the language L. We will construct a deterministic finite automaton $D = (Q', \Sigma, \delta', q_0', F')$ that recognises L.
Let us first consider the easier case where N contains no ϵ transitions, we will later extend this construction to allow ϵ transitions.

1. $Q' = \mathcal{P}(Q)$.
2. For $R \in Q'$ and $a \in \Sigma$, let $\delta'(R, a) = \{q \in Q \mid q \in \delta(r, a)$ for some $r \in R\}$
3. $q_0' = \{q_0\}$
4. $F' = \{R \in Q' \mid R$ contains an accepting state of $N\}$.

This construction allows a deterministic finite automaton to emulate a non-deterministic finite automaton which contains no ϵ transitions.

Equivalence - Proof cont.

To extend this idea to allow ϵ transitions we define the set $E(R)$ to be the set of states that can be reached from members of R by following zero or more ϵ transitions. Note that $E(R) \in Q'$. We replace the previously defined transition function with the following

$$\delta'(R, a) = \{q \in Q \mid q \in E(\delta(r, a)) \text{ for some } r \in R\}.$$

We must also redefine the set q_0'. The redefinition includes all states that are reachable from q_0 via zero or more ϵ transitions.

$$q_0' = E(\{q_0\}).$$

The automaton D accepts a string w if and only if N accepts w. At every step of the computation of D it is in the state that represents the set of possible states that N would be in having read the same portion of input.