Problem Sheet 2

Rowan Saunders

January 17, 2022

1 Convert NFA to DFA Discussion

The following is an NFA over the alphabet $\{h, e\}$, for the language:

$\{w \in \{h, e\} * | w$ is a repeating pattern of he ending in either h or e and accepting the empty string$\}$

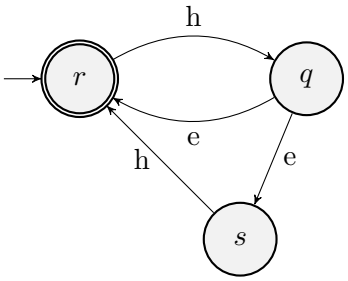

Figure 1: Laughing NFA

ACCEPTS: heh, he, hehehe, heheh

REJECTS: eh, eheh, hheh, heeee

Let $N = (Q, \Sigma, \delta, r, F)$ be a non-deterministic finite automaton, where

- $Q = \{r, q, s\}$
- $\Sigma = \{h, e\}$
- $\delta : Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$
  - $\delta(r, h) = \{q\}$
  - $\delta(r, e) = \emptyset$
  - $\delta(q, h) = \emptyset$
  - $\delta(q, e) = \{r, s\}$
  - $\delta(s, h) = \{r\}$
  - $\delta(s, e) = \emptyset$
- r is the initial state
- $F = \{r\}$

1.1 Convert forum post NFA to DFA

The posted NFA accepts any string from the language $\{0, 1\}$ that starts with 11

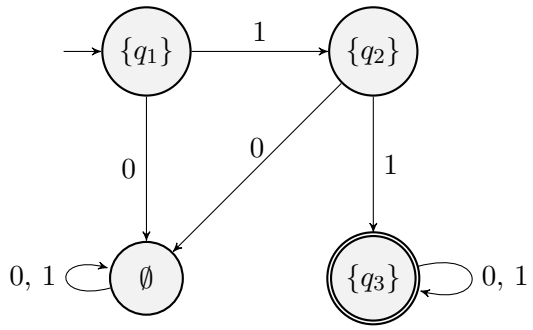

Figure 2: Agnes' NFA

2 Non-Deterministic Finite Automata Notes

As before, a (deterministic) finite automata is a 5-tuple $(Q, \Sigma, \delta, s_0, F)$ where Q is a finite set of states, Σ is the input alphabet, $\delta : Q \times \Sigma \to Q$ is the transition function, $s_0 \in Q$ is the start state, $F \subseteq Q$ is the set of accept states.

For a Deterministic Finite Automata (DFA), all valid states must have a transition function for every member of the input alphabet

For a Non-Deterministic Finite Automata (NFA), the transition function may have duplicate or missing transitions for members of the input alphabet

Formally, the only difference between an NFA and DFA is the transition function.

For a DFA:

$$\delta : Q \times \Sigma \to Q$$

For an NFA we first need to define a mathematical concept called a Power Set

Given a set A, the **power set** of A, denoted $\mathcal{P}(A)$, is the set of all subsets of A.

So, Let $S = \{1, 2\}$

$$\mathcal{P}(S) = \{\emptyset, \{1\}, \{2\}, \{1, 2\}\}$$

Therefore, for an NFA, the transition function is as follows:

$$\delta : Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$$

Where $\mathcal{P}(Q)$ is the power set of Q, and $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ (the alphabet plus the empty string) Therefore, the transition function returns some set of states from Q.

The full formal definition for a Non-Deterministic Finite Automata is a 5-tuple

$$(Q, \Sigma, \delta, s_0, F)$$

Where Q is a finite set of states, Σ is the input alphabet, $\delta : Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$ is the transition function, $s_0 \in Q$ is the start state, $F \subseteq Q$ is the set of accept states.

The following diagram is an example NFA

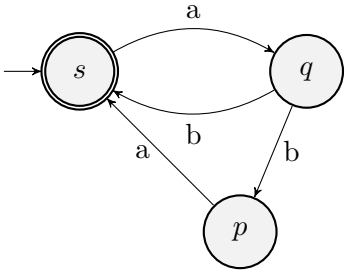

Figure 3: NFA Example

For the FSM in *Figure 3*:

Let $N = (Q, \Sigma, \delta, s, F)$ be a non-deterministic finite automaton, where

- $Q = \{p, q, s\}$
- $\Sigma = \{a, b\}$
- $\delta : Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$
  - $\delta(s, a) = \{q\}$
  - $\delta(s, b) = \emptyset$
  - $\delta(p, a) = \{s\}$
  - $\delta(p, b) = \emptyset$
  - $\delta(q, a) = \emptyset$
  - $\delta(q, b) = \{s, p\}$
- s is the initial state
- $F = \{s\}$

The empty set ($\emptyset$) in the transition function is used to show that those inputs have no combination of state and symbol/arrow, and result in immediate rejection.

The definition of a configuration remains the same for an NFA as it was for a DFA, namely:

> The combination of a state and a string is a configuration. A configuration is a pair: $(q, w) \in Q \times \Sigma*, (q \in Q, w \in \Sigma*)$.

However, the definition of computation for a DFA must be an updated for an NFA

For a DFA the following represents a computation:

> Let (q, w) and (q', w') be configurations, where:
>
> $$w = aw'$$
>
> for some letter $a \in \Sigma$, and
>
> $$\delta(q, a) = q'$$
>
> Then we say that (q, w) yields (q', w') in one step.
>
> For configurations (q, w) and (q', w') we say that (q, w) yields $(q', w'$ if there is a finite sequence of configurations
>
> $$(q_1, w_1), (q_2, w_2), ..., (q_k, w_k)$$
>
> Such that $(q_1, w_1) = (q, w), (q_k, w_k) = (q', w')$, and (q_i, w_i) yields (q_{i+1}, w_{i+1}) in one step for all $i = 1, 2, ..., k - 1$

For an NFA, we replace the definition of "yield in one step" from $\delta(q, a) = q'$ to $\delta(q, a) \ni q'$, such that the new state goes from being equal to the result of the transition function, to being a member of the set of possible states returned from the transition function. Note that a backwards version of the set membership symbol is used purely for symmetry with the old version. This could be written as $q' \in \delta(q, a)$. Additionally, we also need to allow the letter to be include the empty string, by using Σ_ε

So, the full definition for an NFA is:

Let (q, w) and (q', w') be configurations, where:

$$w = aw'$$

for some letter $a \in \Sigma_\varepsilon$, and

$$\delta(q, a) \ni q'$$

Then we say that (q, w) yields (q', w') in one step.

For configurations (q, w) and (q', w') we say that (q, w) yields $(q', w'$ if there is a finite sequence of configurations

$$(q_1, w_1), (q_2, w_2), ..., (q_k, w_k)$$

Such that $(q_1, w_1) = (q, w), (q_k, w_k) = (q', w')$, and (q_i, w_i) yields (q_{i+1}, w_{i+1}) in one step for all $i = 1, 2, ..., k - 1$

The sequence of configurations defined above for an NFA is called a **computation**

The non-deterministic finite automaton $N = (Q, \Sigma, \delta, s_0, F)$ **accepts** the string $w \in \Sigma*$ if (s_0, w) yields (q, ε) where $q \in F$.

Recalling the following definitions:

> We say that the finite automaton M **recognises** the language A if $A = \{w | M \quad \text{accepts} \quad w\}$.
>
> The language **recognised by** a finite automaton M is denoted $L(M)$.
>
> The language A is called regular if there exists some finite automaton M such that $A = L(M)$. (i.e. a finite automaton exists that recognises it)

Two finite automata A and B (deterministic or non-deterministic) over the same alphabet Σ are called **equivalent** if $L(A) = L(B)$, i.e. they accept the same language.

The above definition leads to the following Theorem:

> For every non-deterministic finite automaton A there exists a deterministic finite automaton A' which is equivalent to A.

This Theorem has the following proof:

> Let $N = (Q, \Sigma, \delta, s_0, F)$ be some non-deterministic finite automaton. We construct a deterministic finite automaton $M = (Q', \Sigma, \delta', s'_0, F')$ such that $L(N) = L(M)$.
>
> Let $Q' = \mathcal{P}(Q)$
>
> Let $s'_0 = E(\{s_0\})$, where $E(R) = \{q | q \text{ is reachable from } R \text{ by } \varepsilon \text{ transitions}\}$
>
> Let $F' = \{R \in Q' | R \text{ contains an accept state of } N\}$, Alternatively $F' = \{R \in Q' | R \cap F \neq \emptyset\}$
>
> For $R \in Q', a \in \Sigma$, Let $\delta'(R, a) = \{q | q \in E(\delta(r, a)) \quad \text{forsome} \quad r \in R\}$
>
> Notice $\delta'(R, a) \in Q'$